TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Produktentwicklung

# Automatic Fixture Design Based on Formal Knowledge Representation, Design Synthesis and Verification

**Thomas Christoph Gmeiner**

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

**Doktor-Ingenieurs**

genehmigten Dissertation.

| | |
|---|---|
| Vorsitzender: | Univ.-Prof. Dr.-Ing. Veit Senner |
| Prüfer der Dissertation: | 1. Univ.-Prof. Dr.-Ing. Udo Lindemann |
| | 2. Univ.-Prof. Kristina Shea, Ph.D., Eidgenössische Technische Hochschule Zürich, Schweiz |
| | 3. Assistant Prof. Dr. Farhad Ameri, Texas State University, San Marcos, USA |

Die Dissertation wurde am 09.10.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 22.04.2015 angenommen.

# ABSTRACT

The fabrication of parts in manufacturing systems requires their fixation in a defined position for machining, assembly or inspection processes using fixture devices. Especially for low-volume production and the resulting frequent change-overs, as commonly found in modern manufacturing scenarios, the design and configuration of fixtures have a strong influence on the production time and costs. To address these tasks and increase the flexibility and degree of automation of manufacturing systems, flexible fixture devices and computer-aided or automated fixture design methods are developed. While flexible fixture devices are well established in industry nowadays, no general and commonly used methods, applications or systems for computational fixture design exist so far. This is due to the fact that the involved tasks are wide-spread, knowledge intensive and strongly intertwined with other steps of the design-to-fabrication process. Further, existing computer-aided fixture design systems are rarely integrated with the configuration or embodiment of flexible fixtures on the hardware level.

To address these issues, an automatic fixture design system integrated with a reconfigurable fixture device is presented in this thesis. The system is based on a formal ontology that serves as an integrated knowledge representation and basis for the generation of fixture design candidates through semantic reasoning. A spatial grammar is used for the synthesis of new problem-specific fixture components and design candidates. A tool-fixture interference analysis is used for fixture design verification and a workpiece deformation analysis for evaluation of the design candidates. Prototype software implementations for all methods are developed and their applicability to automate the design and drive the reconfiguration of a vise-type fixture device is evaluated by processing a set of case-study parts.

The results show that the approach and system presented allows for (semi-)automatic fixturing of parts of different and non-predefined geometries and, thus, for an increase of the degree of flexibility and automation of a manufacturing system. The work further contributes to the areas of ontological fixture design knowledge representation, the use of spatial grammars for detailed engineering design tasks, the automation of cutting tool interference analysis based on NC codes and the automation of finite element analysis in the area of fixture design verification.

# ZUSAMMENFASSUNG

Zum Durchlaufen von Fertigungs-, Zusammenbau und Inspektionsprozessen in Produktionssystemen müssen Bauteile jedweder Art mit Hilfe von Spannvorrichtungen in einer referenzierten Lage fixiert werden. Besonders im Angesicht von modernen Produktionsszenarien, wie der automatisierten Fertigung von Kleinserien oder Einzelteilen, und den damit einhergehenden häufigen Umrüstvorgängen, haben der Entwurf und die Konstruktion von Spannvorrichtungen einen großen Einfluss auf die Produktionskosten. Um den Automationsgrad und die Flexibilität von Produktionssystemen mit Hinblick auf diese Aufgaben zu erhöhen, werden flexible Spannvorrichtungen und computer-unterstützte oder automatisierte Methoden zum Entwurf von Vorrichtungskonfigurationen erforscht. Während sich flexible Spannvorrichtungssysteme industriell etablieren konnten gibt es bis dato keine generell anwendbaren Methoden, Anwendungen oder Softwaresysteme zum Erstellen und Verifizieren von Vorrichtungsentwürfen. Dies basiert darauf, dass die involvierten Prozesse stark unterschiedlich, wissensintensiv und hochgradig vernetzt mit anderen Produktionsprozessen sind. Die existierenden computer-basierten Systeme sind des Weiteren nur sehr selten mit der physikalischen Konfiguration oder Anpassung einer flexiblen Vorrichtung integriert.

Als Möglichkeit zur Lösung dieses Problems wird in dieser Dissertation ein automatisches System zum Spannvorrichtungsentwurf, dass mit der Konfiguration einer flexiblen Spannvorrichtung für die spanende Fertigung gekoppelt ist, vorgestellt. Das System nutzt eine formelle Ontologie als integrierte Wissensrepräsentation und Basis zur automatischen Erstellung von Entwurfskandidaten durch semantische Schlussfolgerungen. Außerdem werden eine räumliche Grammatik für die Synthese von neuen, problem-spezifischen Vorrichtungskomponenten, eine Methode zur Analyse von Interferenzen zwischen der Vorrichtung und den Werkzeugen einer Fertigungsmaschine als Basis für die Entwurfsverifikation und eine Methode zur Analyse der Werkstückdeformation als Basis für eine Bewertung verschiedener Entwürfe entwickelt und präsentiert. Prototypische Softwareumsetzungen für alle Methoden werden vorgestellt und ihre Anwendbarkeit zur Automatisierung des Entwurfes und der Konfiguration einer flexiblen, schraubstockartigen Spannvorrichtung wird anhand mehrerer realistischer Beispielwerkstücke evaluiert.

Die Ergebnisse zeigen, dass die Ansätze und das Gesamtsystem das (semi-)automatische Spannen von Werkstücken unterschiedlicher und nicht vordefinierter Geometrien ermöglicht und so zur Steigerung der Flexibilität und des Automationsgrades eines Fertigungssystems beitragen kann. Des Weiteren trägt die Arbeit zur ontologischen Darstellung von Spannvorrichtungswissen bei, ist ein seltenes Beispiel für den Einsatz von räumlichen Grammatiken für die computer-basierte Synthese von detaillierten Entwürfen eines stark restringierten Designproblems, präsentiert einen Ansatz zur Automation von Kollisionsanalysen mit Maschinenwerkzeugen und zur Automation von Finite-Elemente Berechnungen im Bereich des Vorrichtungsbaus.

# ACKNOWLEDGEMENT

Without the help and assistance of several people the completion of this thesis would have been impossible or at least much harder. I want to thank my professors, my former colleagues at the Institute of Product Development at TUM and at the Engineering Design and Computing Laboratory at ETHZ, the students that have worked with me and my family.

First, I want to thank Prof. Dr. Shea for accepting me as a Ph.D. candidate in her group, for the invested time, the many fruitful discussions and especially for her flexibility that allowed me to stay and finish my thesis at TUM while the group moved to ETHZ, which made it much easier to combine family and work for me.

My gratitude also goes to Prof. Dr.-Ing. Lindemann for welcoming me at the Institute of Product Development, providing the necessary infrastructure to be productive and for doing everything in his power to make the defense of my thesis possible.

Special thanks go to Prof. Dr. Ameri for the collaboration and his valuable input on the topic of ontologies and for his effort as third examiner. His hospitality during my stay at Texas State University and the many joint activities, lunches and coffee breaks made my research visit in the US not only productive but a real pleasure.

My former colleagues made my time as research assistant a lovely and unforgettable experience that I would not want to miss. Among all the great people that I worked with I would especially like to name Dr.-Ing. Christoph Ertelt, who served as my mentor and taught me a lot about the "research game", Dr.-Ing. Frank Hoisl, who helped me with the use of his software Spapper, Dr.-Ing. Bergen Helms for the discussions and insight into the world of formal representations, Clemens Münzer, who provided me with a home abroad in Zurich as well as Dr.-Ing. Torsten Metzler and Wolfgang Bauer for the relaxing conversations during the final writing phase.

In addition I would like to thank all the students that have worked with me on the thesis related topics and, thus, helped me in finishing this thesis.

I am indebted to my parents and my brother for supporting me and all of my ideas throughout my life and want to thank them for it. And, finally, my biggest thanks go to my girlfriend and my kids for being who they are, for believing in me and for taking my mind off work when possible.

Munich, May 2015 Thomas Gmeiner

# PREVIOUS PUBLICATIONS

The following publications are part of the work presented in this thesis:


GMEINER & ERTELT 2010
: Patent WO/2010/130590 A4 (Publication: 18.11.2010). Technische Universität München. 2010055959 -Gmeiner, T.; Ertelt, C.: Method and Device for Holding and Machining a Workpiece.

SHEA et al. 2010
: Shea, K.; Ertelt, C.; Gmeiner, T.; Ameri, F.: Design-to-fabrication automation for the cognitive machine shop. Advanced Engineering Informatics 24 (2010) 3, p. 251-268.

GMEINER & SHEA 2011
: Gmeiner, T.; Shea, K.: Development of an Ontology for the Reconfiguration of a Vise-Type Fixture Device. In: ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Washington, D.C., USA, 28-31 August 2011. p. 251-261. ISBN: 978-0-7918-5482-2.

GMEINER & SHEA 2013a
: Gmeiner, T.; Shea, K.: An Ontology for the Autonomous Reconfiguration of a Flexible Fixture Device. Journal of Computing and Information Science in Engineering 13 (2013) 2, p. 11.

GMEINER & SHEA 2013b
: Gmeiner, T.; Shea, K.: A Spatial Grammar for the Computational Design Synthesis of Vise Jaws, ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE 2013). Portland, Oregon, USA, August 4-7 2013.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AFD | Automated Fixture Design |
| CAD | Computer-Aided Design |
| CAFD | Computer-Aided Fixture Design |
| CDS | Computational Design Synthesis |
| CogMaSh | Cognitive Machine Shop |
| COS | Coordinate System |
| DoF | Degree of Freedom |
| FD | Fixture Design |
| FE | Finite Element |
| FEA | Finite Element Analysis |
| FFS | Flexible Fixture System |
| FMS | Flexible Manufacturing System |
| GUI | Graphical User Interface |
| IWS | Initial Working Shape (of a spatial grammar rule) |
| LHS | Left-Hand Side (of a spatial grammar rule) |
| MFFS | Modular Flexible Fixture System |
| NC | Numerical Control |
| OWL | Web Ontology Language |
| RHS | Right-Hand Side (of a spatial grammar rule) |
| SGI | Shape Grammar Interpreter |
| SWRL | Semantic Web Rule Language |
| TSV | Tool Swept Volume |
| WP | Workpiece |
| XML | Extensible Markup Language |

# 1. Introduction

Today's manufacturing industry is facing new challenges imposed by the modern market demands for customized products and decreasing lead time (LINDEMANN & BAUMBERGER 2006, pp. 7-9). For the manufacturers, this causes more variants to be produced, the frequent introduction of new parts, low-to-medium volume production and frequent change-overs (KAMARTHI et al. 2009). To compete in such a scenario, a high flexibility in manufacturing is necessary throughout the whole process from a design to the finished product. Manual shop floors, making use of human labor and intelligence, can provide the required flexibility. However, considering current globalized markets they are only economically efficient in low-wage countries. In high-wage countries, on the other hand, increasing the level of automation and flexibility of manufacturing systems is the common approach to stay competitive. Despite tremendous research efforts in the last few decades, the degree of flexibility and automation of modern manufacturing systems for low volume production is still not sufficient to meet the described market demands in an economical way (ELMARAGHY 2005).

## 1.1 Automating the Design-to-Fabrication Process

While modern computerized numerically controlled machines (CNC) and robots integrated in flexible manufacturing systems (FMS) or reconfigurable manufacturing systems (RMS) allow for a fast machining and handling of parts, many tasks required in the process to get from the design of a part to the point of fabrication still heavily rely on the knowledge, reasoning and planning capabilities of human experts.

The steps from a digital model of a part to the fabricated, physical product can be summed up under the term design-to-fabrication process. These steps include, but are not limited to, setup planning, machining planning, fixture design, production planning and resource planning. Most of the steps include several tasks or sub-steps that need to be carried out and are intertwined with other process steps, making the design-to-fabrication process complicated and knowledge intensive. For many of these tasks different computer-aided approaches exist that are known under terms like computer-aided manufacturing (CAM) and computer-aided process planning (CAPP) or under the higher-level term computer-integrated manufacturing (CIM). However, these tools only aid an expert user in the design-to-fabrication process. Approaches to automate the tasks exist but are generally limited to a certain geometric part spectrum or manufacturing system, require pre-programming and cannot flexibly react to changes or deal with non-predefined part geometry (ERTELT 2012, pp. 1-20). Addressing this issue is the driver for the Cognitive Machine Shop concept (CogMaSh), to which the presented work contributes.

## 1.2 The Cognitive Machine Shop Concept

The Cognitive Machine Shop concept addresses the needs required for autonomous manufacturing of non-predefined parts in low-volume or one-of-a-kind production by enhancing a manufacturing system with increased degree of automation and flexibility, taking both software and hardware aspects into account (SHEA 2010). This is done by (1) embedding

cognitive capabilities, like perception, reasoning and planning, in the design-to-fabrication process, the production planning process and the control systems (SHEA et al. 2010; BANNAT et al. 2011; ZAEH et al. 2010) and by (2) developing or extending the functionality of existing flexible manufacturing hardware (GMEINER 2008; ERTELT et al. 2009).

The aim is manufacturing systems that, once set up, can flexibly produce parts with a non-predefined geometry with no or minimal user interaction on the basis of only a 3D model of the part to be produced. This means that the tasks of the design-to-fabrication process that are so far carried out by humans need to be automated. To allow for this, the system components need to be aware of their capabilities and environment, be able to communicate, plan and to adapt to unforeseen events or problems. Knowledge models that allow for reasoning, decision-making and planning as well as generative design generation and verification approaches play a crucial role in realizing this vision. To validate any approaches and prototypes developed in the CogMaSh project, they are integrated and tested in a demonstration manufacturing system consisting of an automated storage, a conveyor system, handling robots, an assembly station, a CNC lathe and a vertical 3-axis CNC milling machine. Figure 1-1 shows the setting of the CogMaSh demonstrator system.

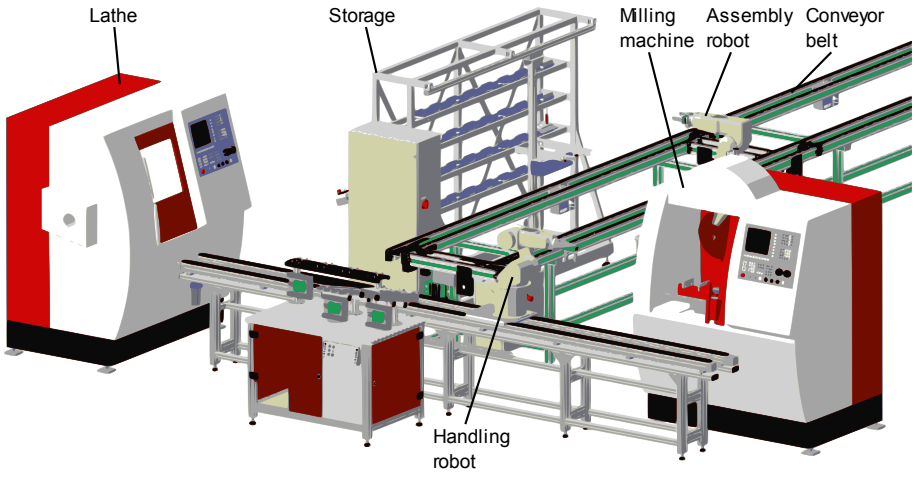



**Figure 1-1 Cognitive Machine Shop with main hardware components highlighted**

Starting with a 3D part model to be produced, the first steps of the design-to-fabrication process include the selection of a raw part or workpiece to machine the part from, the setup planning, planning of the machining process, including machine and cutting tool selection and tool path generation, and communication of this information within the system to drive the physical fabrication, e.g. to retrieve the raw material from the storage or check whether the machine and cutting tools are available. Within the CogMaSh research project an ontological representation of workpieces that allows for reasoning about and ranking of possible raw parts available in the storage based on geometric and material requirements was developed (SHEA et al. 2010). Further, an approach for automated machining planning for the CNC milling machine based on a spatial grammar and search algorithms was developed (ERTELT 2012; ERTELT & SHEA 2009). For a given setup, the approach allows for the generation of fabrication plans in the style of NC

codes. This enables machining planning for non-predefined parts based on modeling fundamental machine capabilities.

Using these two approaches, a raw part can be selected and the NC code for the machining of one or multiple setups required for the complete machining of a part can be generated within certain restrictions. The information can be transferred to the CogMaSh system and, in combination with a decentralized, RFID-controlled local planning system (ZAEH et al. 2012), used to drive the retrieval of the raw part from the storage, transport it to the milling machine via the conveyor system and handling robot and to numerically control the milling process. This scenario, however, raises a problem on the hardware side. The milling machine, with its NC controlled feed axis and cutting tool changer, offers the necessary flexibility to machine different part shapes and remove volume as needed from the raw part. Other components in the system, such as the robot grippers, the transport pallets or the fixture device required to hold the raw part during machining, do not offer the necessary flexibility to process parts of non-predefined geometry. As the degree of flexibility and automation of a manufacturing system always depends on that of each single component in the system, this is a critical issue. Flexibly automating all other process steps is of little help, if a human worker is still required to configure grippers or fixture devices every time a new workpiece (WP) geometry is processed.

To enable the necessary flexibility and degree of automation on the hardware level, the mentioned components should autonomously adapt to different part shapes, e.g. through self-adaption or external reconfiguration. Achieving such autonomous adaption for a fixture device is the topic of this work. This requires flexible fixture hardware as well as approaches for reasoning about possible and feasible fixture designs, fixture planning, computational design generation and verification. Due to the close relation, the methods developed and findings are also interesting for the area of robot grasping or the adaption of transport pallets.

## 1.3 Workholding Fixtures

As presented, workholding fixture devices, or fixtures, play a crucial role in increasing the degree of automation and flexibility in customized manufacturing scenarios and manufacturing systems in general. Fixtures are required to accurately locate, hold and support a part for machining, assembly, inspection and other operations throughout the manufacturing process (NEE & TAO 2004). Fixtures and their design and fabrication process have a high impact on the production costs and the lead-time. GANDHI & THOMPSON (1986) state that dedicated fixtures make up 10-20% of the total system costs. In low-volume production scenarios with a wide variety of part shapes and frequent change-overs, aspects of flexible workholding and efficient fixture design become even more important. Due to the high impact on the manufacturing costs, the topic has long been recognized by industry and researchers alike. Two basic ways to address the issue emerged from this, the development and use of:

- flexible fixtures that can adapt or be reconfigured to fit different part shapes and
- computer-aided fixture design (CAFD) systems that help to improve and speed-up the fixture design and verification process.

Flexible fixture systems (FFS), as opposed to dedicated fixtures, can be used to hold different parts or part variants. FFS reduce the fixturing costs and lead-time up to 80% (HARGROVE & KUSIAK 1994; KANG & PENG 2008). Different types of FFS exist that can be classified into single-structure and modular flexible fixtures (BI & ZHANG 2001, p. 2874). Single-structure flexible fixtures have a fixed base structure but can adapt to different part shapes, e.g. by using phase-changing materials or hydraulic pin-array clamps. Their use often results in a non-referenced positioning of the part, making a surface or edge detection necessary before machining. Modular flexible fixture systems (MFFS) are built from modular components, such as locators or clamps, mounted onto a T-slot, dowel-pin or threaded-hole base-plate. MFFS have become a de-facto standard for the use in modern manufacturing systems because of their high flexibility.

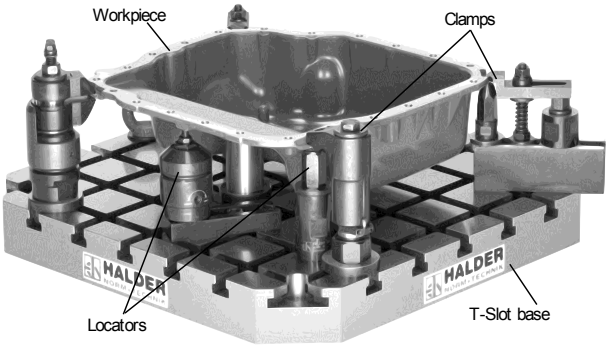



**Figure 1-2 Modular T-slot fixture with locators, clamps and workpiece (picture © D. MENGEMANN SPANNTECHNIK GMBH 2014)**

Figure 1-2 shows a modular fixture with a T-slot base carrying several locators and clamps used to clamp a workpiece. With MFFS a fixture is designed and assembled for each WP, the WP is clamped in the fixture and, in case several manufacturing processes are required, the complete assembly is handled as one entity in the manufacturing system to maintain the WP position. In case multiple WPs are to be machined, it is common practice to build a fixture for each WP to speed up the cycle time, as the loading and unloading of the fixture can be time consuming. Despite the high flexibility, MFFS have several drawbacks: they are cost intensive, as a big stock of fixture modules is necessary, which is often not used to capacity, and as the assembly, loading and unloading of the fixtures is mostly done manually. Passing the WP including the fixture increases the weight that has to be handled, making stronger robots and transport systems necessary in the FMS. Further, the design and assembly of MFFS requires significant expert knowledge and experience (BI & ZHANG 2001). To address these shortcomings, a different flexible fixture device was developed as part of the CogMaSh concept.

## 1.4 The Flexible Vise

For the CogMaSh system the aim is to develop a flexible fixture system for WP machining that can be fully automated, including the reconfiguration/adaption process to non-predefined WP shapes. The system has to circumvent the shortcomings of commercially available flexible

fixtures, such as high costs, high weight or unreferenced positioning of the WP. The result of the systematic development is the flexible vise presented in (GMEINER 2008; ERTELT et al. 2009). For a practical evaluation, the fixture is installed in the 3-axis milling machine of the CogMaSh system. A CAD model of the vise and a photo of the prototype are shown in Figure 1-3.

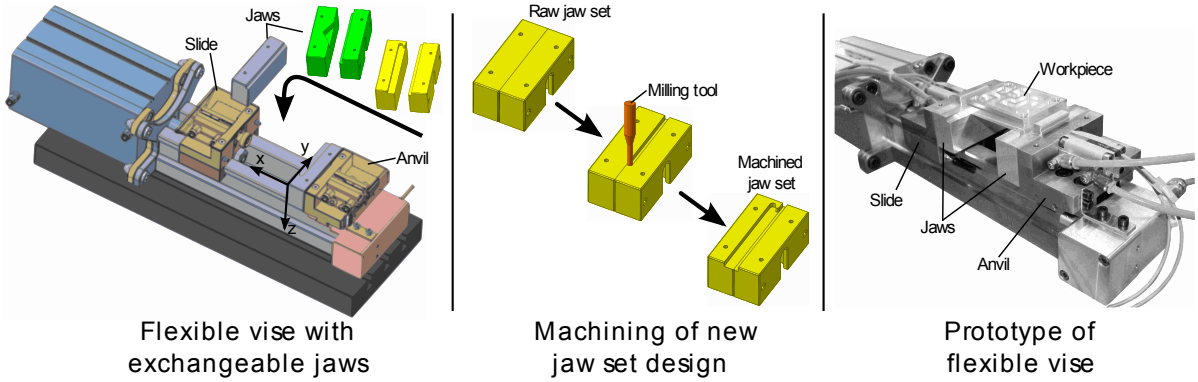



**Figure 1-3 The flexible vise: CAD model showing exchange of jaws and coordinate system of vise (left), adaption of jaw set through machining (middle) and picture of prototype (right)**

The flexible vise is a pneumatically driven, single-acting vise, i.e. a vise with a fixed anvil and a moveable slide, that can be reconfigured and adapted to hold different WP geometries. Vises have the advantage of being versatile fixtures and having short set-up times (JONEJA & CHANG 1999, p. 657). The reconfiguration of the vise is done by exchanging the vise jaws. The jaws are the fixtures' components that are in contact with the WP and serve as locating, supporting and clamping entities. In the FMS, the jaws are handled like WPs. They can be stored, transported and handled by the existing robots. To allow for their exchange, they can be attached or detached to the anvil and slide of the vise by actuators with a (re-)positioning accuracy of ±0.02mm. By using jaws with different jaw-WP contact surface geometries, both prismatic and cylindrical WP geometries can be accurately located, supported and clamped. Besides the reconfiguration, the vise can also be adapted to WP geometries by creating the required contact surfaces through machining of a pair of raw jaws. This machining process can be done in place, i.e. with the milling machine the vise is installed on and with the jaws attached to the vise. This is shown in the middle of Figure 1-3. For the machining, the jaws have to be pressed against each other to increase the positioning accuracy of the slide.

To automate the fixture process, the control system of the vise can set the clamping force exerted by the slide and the clamping width, i.e. the distance between the anvil and slide jaw, both in a closed loop control. The clamping force range is defined by the pneumatic pressure in the system and the minimum pressure required for accurate functioning of the pneumatic piston. The clamping width range is defined by the geometry of the vise. Further, sensors are used to determine whether the jaws are correctly attached to or detached from the anvil and slide and whether a WP is correctly clamped. The vise control system, based on a programmable logic

controller, is intertwined with the FMS control system to be able to communicate with the machine and the handling robot.

## 1.5 Research Goals, Scope and Assumptions

The vision for the flexible vise is to allow for the complete automation of the WP fixture process in the CogMaSh milling machine by combining an automatic fixture design system (AFD) with an automatically reconfigurable/adaptable fixture. In a first usage scenario of the flexible vise, a set of different jaws is added to the storage of the CogMaSh and for every WP and setup to be machined the specific jaw set is given as part of the local manufacturing plan. This, however, requires that a human designer manually creates a fixture design (FD), i.e. picks a feasible jaw set design from the existing ones or generates a new jaw set design, defines the clamping width and clamping force and adds the information to the hard-coded manufacturing plan. Thus, this process is far from full automation. To further increase the degree of automation of the fixture device and the overall FMS, approaches to flexibly automate these tasks are required.

Based on the 3D model of a new, non-predefined part to be produced, a geometrically matching set of jaws has to be determined from the list of existing jaw designs or a new matching set of jaws has to be generated. Further, a feasible clamping length and clamping force has to be defined. The automatically generated fixture design should then be used to drive the reconfiguration or adaption process of the flexible vise on the hardware side. All this should happen computer-based with no or minimal user-interaction.

The work presented in this thesis focusses on the following aspects, which should allow for making full use of the flexible vise's capabilities:

- Representation and storing of fixture designs and vise configuration data in a formal, computer-interpretable fashion that can be used for communication within the FMS;
- Reasoning about possible fixture designs that reuse the existing vise jaws;
- Automatic generation of new jaw set designs that fit a given WP and can be manufactured on the CogMaSh milling machine;
- Verification of the feasibility of the designs selected through reasoning or generated automatically;

The research goal is to develop an approach and a software prototype for each of these aspects, which allows for automation and inter-operation among the methods. All software prototypes are to be based on open-source software tools to facilitate reuse and extension by others. The applicability of the methods has to be validated based on a set of realistic workholding jobs, i.e. parts to be produced. As shown before, other manufacturing system components besides workholding fixtures require autonomous adaption or reconfiguration to allow for the processing of non-predefined parts. Thus, the study and development of requirement-driven approaches for tasks like formal design representation, knowledge-based reasoning, (re-)configuration planning or design generation and verification are not only important for or limited to the area of workholding fixtures but may be applied to other system components to increase the degree of automation and flexibility of a complete manufacturing system. The

flexible vise can, therefore, be seen as an application and evaluation scenario for the methods and approaches developed.

The scope of the AFD system is to identify or generate a feasible FD for any given workholding job, if such a design is possible. Existing fixture components and designs should be reused whenever possible to minimize the need for new fixture components. Optimality of the generated designs is not intended, nor the improvement of the FD accuracy or the duration of the FD process. Aspects of fixture accuracy, such as WP tolerances and datums are, thus, not considered. Once the basis of the AFD system exists, it may be extended to include these aspects. Although setup planning is considered as one of the four main tasks of fixture design (see Section 2.2), it is excluded from this research because it would exceed the scope and can be seen as a stand-alone task. As an existing setup plan is necessary for FD, the setup planning is carried out manually as described in Section 3.8.

Based on the capabilities of the vise, the three-axis milling machine, the handling robot and the conveyor system of the FMS, the WP shapes and dimensions that can be processed are limited. Only prismatic and cylindrical WP shapes or a combination of both, sometimes called "generalized polyhedra" (WALLACK & CANNY 1996), can be handled and are, thus, considered in the presented approach. The restriction to these WP shapes is common for both modular (WU et al. 1998b) and vise-type fixtures. Still, a wide variety of WP geometries is possible with these restrictions as shown in Section 3.8.

## 1.6 Structure of the Thesis

The structure of this thesis is presented in Figure 1-4. The motivation for the work, sparked by the CogMaSh research project in combination with the economic relevance of automatic workholding and fixture design, is given in the first section. Further, the research goals and the scope of the work are presented.

In Section 2 basic fixture design terminology, related work in the area of FD and the background to the applied methods is presented. The tasks involved in FD and approaches to computationally solve them, as presented in CAFD and AFD literature, a definition of the term and the use of ontologies for engineering purposes and the basics of CDS with a focus on spatial grammars and their application in engineering are discussed.

In the third section, the research objectives are derived based on the shortcoming identified in the related work. The overall approach for the presented work is laid out and the selection of methods is explained. The structure of the automatic fixture system and the FD process using the system is shown. The ontological knowledge representation, fixture configuration design based on reasoning, computational fixture component synthesis using a spatial grammar, interference analysis for FD verification using tool-swept volume models and FE-based workpiece deformation analysis for evaluation of the design candidates are introduced. In the end, the WPs and setups used as case-studies are briefly introduced.

Section 4 presents the AFD ontology in detail, including the development methodology, how the domain of interest is covered, defined and formalized. The resulting ontology structure and the formal representation of FDs is discussed before the abilities to reason about modeled knowledge for fixture configuration design is explained and evaluated with case-study WPs.

The spatial grammar for the CDS of fixture components is presented in Section 5. Details on the grammar development, the shape grammar interpreter used, the design requirements, their embedding in the grammar and their effect on the grammar vocabulary and rules are explained. The idea of annotating the 3D WP models to include setup planning knowledge is introduced and the grammar is verified by generating FDs for case-study WPs and setups.

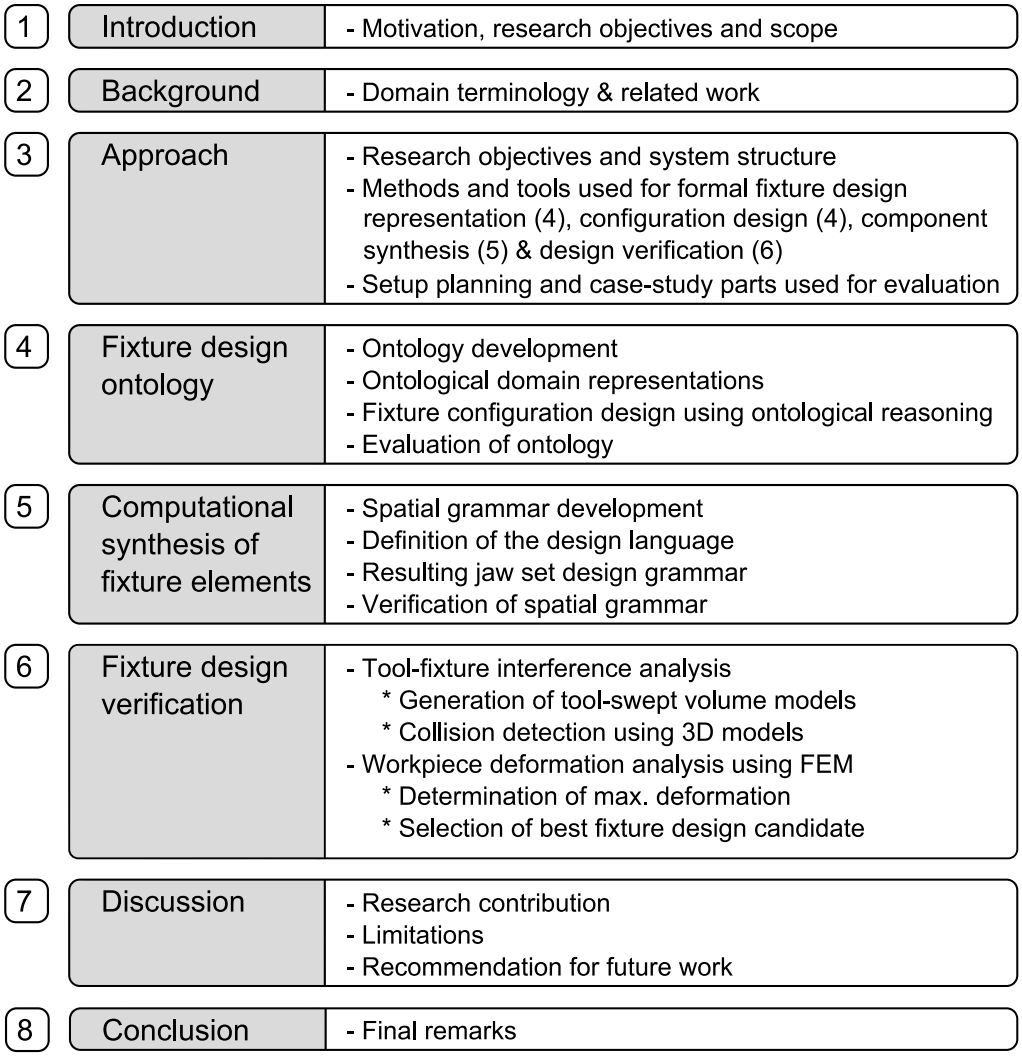



**Figure 1-4 Structure of the thesis**

In Section 6 the structure and development of the cutting tool-fixture interference analysis for FD verification is presented. The automated generation of assembly models of the FD candidates identified by ontological reasoning, the extraction of machining data from the NC files, the generation of tool-swept volume models and the interference test itself are laid out before the approach and implementation is verified using some case-study setups. Further, the FEM-based analysis of the WP deformation caused by the machining and clamping forces and used for evaluating the feasible design candidates is presented. The FEA model definitions are

laid out and the approach is evaluated using several FD candidates of two case-study setups. The aspects of automating the analysis using scripts are discussed.

A final discussion of the research contributions and capabilities of all methods and applications, the limitations of the AFD system and some recommendations for future work are given in Section 7.

Finally, a short conclusion is presented in Section 8.

# 2. Background

In this section basic fixture design terminology, as used throughout this work, is defined. Further, related work in the areas of computer-aided fixture design (CAFD), ontologies for formal engineering knowledge representation and computational design synthesis with a focus on spatial grammars is presented. As this work deals with the design and configuration of a specific flexible fixture device, dedicated fixtures are not further considered. Further, no review of different types of flexible fixtures is included for the same reason. The interested reader can find a survey of flexible fixture systems in BI & ZHANG (2001).

## 2.1 Basics of Fixture Design

Fixture design (FD) describes the process of generating a feasible fixture for a given workholding job or fixture problem, i.e. a workpiece (WP) to be held, considering the part design and process requirements. The aim of all fixtures is to accurately position or "locate" a WP for a manufacturing process, such as machining or assembly, and to restrict the degrees-of-freedom (DoF) of the WP. Locating is realized via contacts between the WP and the components of the fixture. To ensure that the WP stays located, i.e. in contact with the locators, when it is loaded with forces, such as machining forces, clamping components, also called clamps, are used to press the WP against the locators. To prevent the WP from excessive deformation through the machining or clamping forces the fixture may require additional supporting components, also called supports (BOYLE et al. 2011).

### 2.1.1 Locating Principles

The process of positioning the WP via contacts at different surfaces is called locating. Several different locating principles are described in literature (HOFFMAN 2004, p. 21ff). The 3-2-1 locating principle and the V-block locating principle are the most common for outside surface locating of generalized polyhedral WPs, and thus also the most common for modular or vise-type fixtures.

The 3-2-1 locating principle uses three contact points in one plane, two contact points in a plane perpendicular to the first and one contact point in a plane perpendicular to the first and not parallel to the second to accurately locate a WP. The locating principle is depicted in Figure 2-1. In the presented work, the first plane is called the primary locating plane, and the second and third plane are called the secondary and tertiary locating planes. An unrestricted WP has six DoF with 12 possible movement directions, six translational movement directions along the three axes (X,Y,Z) and six rotational movement directions around the axes (Rot X, Rot Y, Rot Z). A WP located with the 3-2-1 locating principle only has three translational planes of movement remaining. In the shown example this would be the translations in positive X-direction, positive Y-direction and negative Z-direction. A fixture that follows the 3-2-1 principle, therefore, does not provide form closure of the located WP. Thus, clamps that push the WP against the locators are required to fully constrain the WP. Depending on the number of clamps used and on how many of the remaining movement directions they directly restrain,

the use of clamps leads to a form closure (WP clamped in x-, y- and z-direction) or a force closure (one or two clamping directions missing). A FD with force closure relies on the reaction forces at the contacts, especially the friction forces, to withstand any external loads, such as the machining forces.

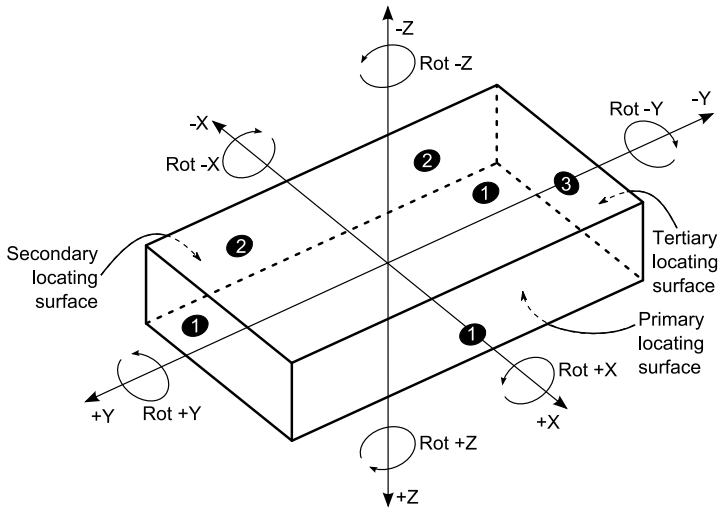



**Figure 2-1 3-2-1 locating principle restricting the degrees of freedom of a prismatic workpiece (based on HOFFMAN (2004, p. 25))**

The V-block locating principle uses two contact surfaces that build a V-shape to locate a WP via two line (or point) contacts, as illustrated in Figure 2-2. The V-block locating principle is best suited to locate curved or cylindrical WP surfaces and can be used for horizontal or vertical locating. However, V-block locating can also be successfully applied to irregular WP shapes with complex outer contours (HOU & TRAPPEY 2001). The two contacts established at the sloped surfaces of the V-block are the secondary locating contacts. Additionally, a primary surface or three-point contact at a plane perpendicular to the two sloped surfaces is required to fully locate the WP. A WP located with the V-block locating principle only has two translational and two rotational movements remaining. In the example shown in Figure 2-2, this would be the translations in the positive X-direction and the negative Z-direction plus the rotations (positive and negative) around the Z-axis. The use of a single V-block locator, therefore, does not lead to form closure of the located WP. For full constraint of all WP movements, clamps that push the WP against the locators are required. If form or only force closure can be established depends on the WP shape. For WP with both cylindrical locating and clamping surfaces, only force closure for the Z-rotation can be achieved. If a planar or irregular clamping surface is used, form closure can be achieved in the primary locating plane (x-y plane in Figure 2-2).

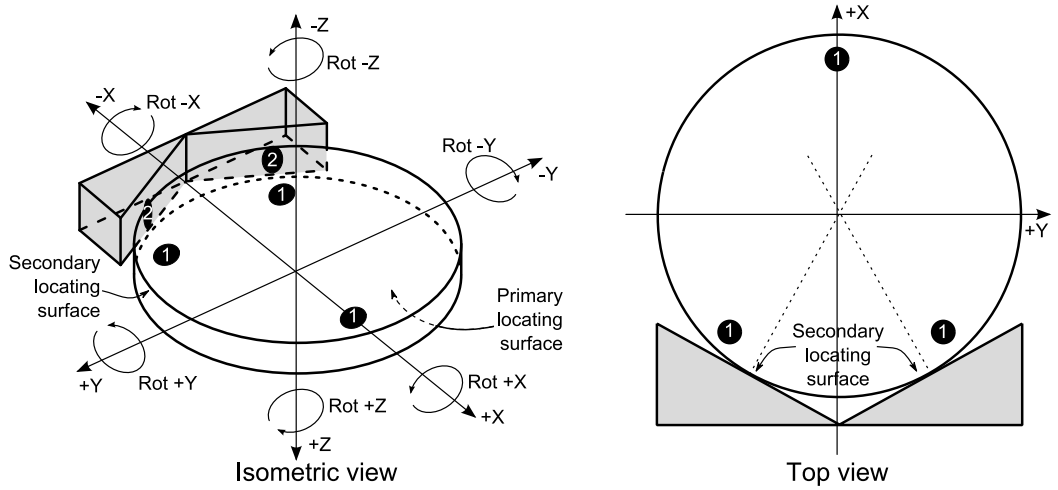



**Figure 2-2 V-block locating principle restricting the degrees of freedom of a cylindrical workpiece**

The jaws of the flexible vise can follow the 3-2-1 and the horizontal V-block locating principles. Vertical V-block locating is not possible as such jaw shapes cannot be machined with the flexible vise. As common for machine vises, surface or line contacts are used instead of point contacts. For the 3-2-1 locating principle the three contacts in the primary locating plane are established as two surface contacts, one on the anvil jaw and one on the slide jaw. This way, the distance between the locating "points" is maximized which is beneficial for the stability of the locating (HOFFMAN 2004). Following common fixture design terminology, the horizontal WP contact surface at the locating jaw is called the *primary locating surface* and the horizontal contact surface at the clamping jaw the *supporting surface*. The other surfaces used for locating are called the *secondary* and *tertiary locating surfaces* and establish either two surface contacts or a surface and a line contact to the locating jaw. For the V-block locating principle only one (cylindrical) WP locating surface is used that establishes two line contacts to the V-block. Thus, the *secondary* and *tertiary locating surface* is the same. The WP surface that is subject to the clamping force is called the *clamping surface* and the main surface of the WP to be machined in a given setup is called the *active surface*. As the flexible vise is installed on a 3-axis milling machine with vertical tool approach direction, the *active surface* is generally the top surface of the WP.

When the 3-2-1 locating principle is used, the three remaining possible movements are restricted by the clamping force exerted by the slide jaw. The clamping force, however, only acts perpendicular to the secondary locating surface, for example in the negative X-direction in Figure 2-1. Clamping the WP against the primary and tertiary locating surfaces is not common with vise-type fixtures as it requires additional clamps. Instead, the frictional forces between the jaws and the WP caused by the clamping force have to ensure that the two remaining movement planes are restricted. The V-block locating with the flexible vise also relies on force closure to withstand any WP movements in negative Z-direction or, in case both the WP locating and clamping surfaces are cylindrical, around the Z-axis.

### 2.1.2 Fixture Design Process

The process of generating a FD includes several tasks that can be structured or clustered into phases. Two slightly different models of the FD process are presented in literature: the model by RONG et al. (2005, p. 11ff) and the model by BI & ZHANG (2001). In this work, the model presented by RONG et al. (2005) in an extended version presented by BOYLE et al. (2011) is used, because of the stricter differentiation of the phases and the explicit inclusion of fixture requirement definition. The model divides FD into four main phases: setup planning, fixture planning, fixture configuration design and fixture design verification (see Figure 2-3). Each task requires certain input data and generates output data required for the subsequent tasks or the final fixture configuration. The input data can be manifold, as many aspects affect the FD process. Geometry, tolerance and material data is required as well as machining process and manufacturing capability data. The outcome of the overall fixture design process is generally an informal or formal fixture design in the form of a FD model, a fixture component list, an assembly plan and the required clamping force magnitudes. Considering the design-to-fabrication process, FD is considered to be part of the process planning stage (CAPP) that happens after the design of the part (CAD stage) (RONG et al. 2005; ERTELT 2012). Additionally, the manufacturing planning (CAM) can have an influence on the FD process, e.g. when the tool paths for the machining of a setup are taken into account.

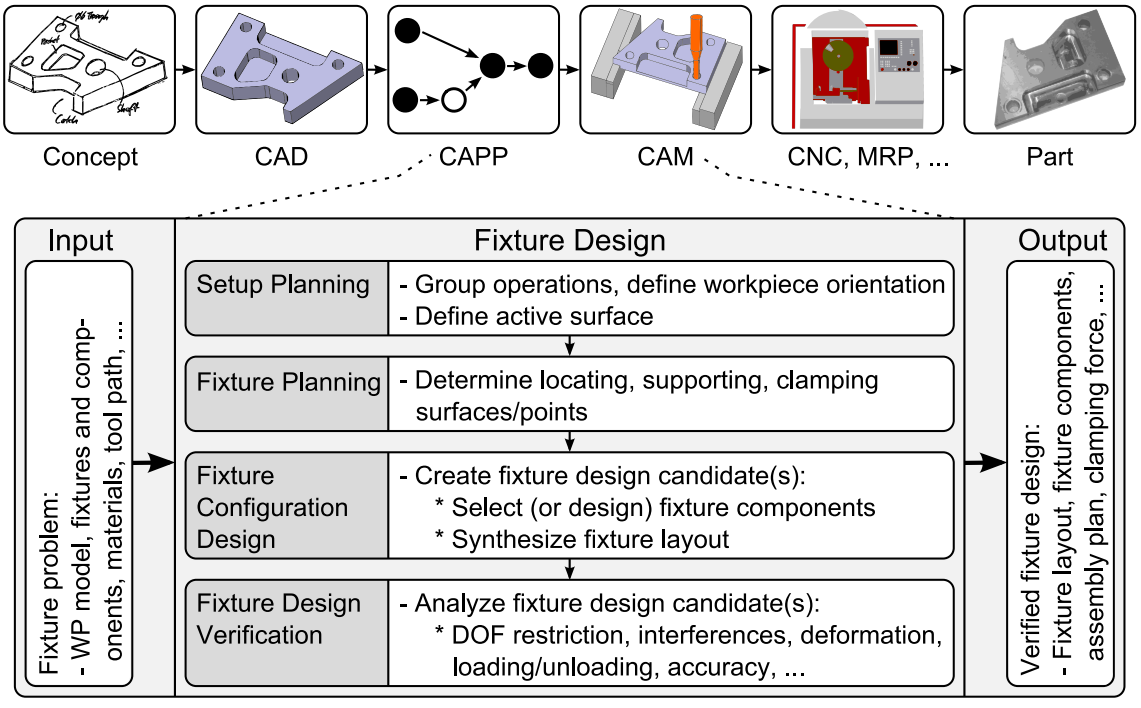



**Figure 2-3 Fixture design (tasks) as part of the design-to-fabrication process**

Setup planning deals with identifying and grouping features of the WP that can be machined in one WP orientation, i.e. without altering the reference position of the WP, and with defining a

sequence of setups that allows for the complete machining of all WP features. Thus, every setup defines a basic WP orientation, the active surface and the datum surfaces used for locating.

In the fixture planning stage the fixture requirements are defined. The fixture requirements must include basic physical requirements and can further include e.g. constraining or collision prevention requirements (BOYLE et al. 2011, p. 3). Physical requirements deal with the ability of the fixture to physically accommodate the WP shape and allow access to it. Constraining requirements relate to ensuring WP stability and a limitation of the WP and fixture deformation caused by the machining and clamping forces. Collision prevention requirements focus on the fact that the cutting tools should not interfere with the fixture or that the fixture components should not interfere with each other. Often fixture requirements are conflicting and it is up to the designer to decide on their importance with regard to the given fixture problem and environment.

Besides defining the requirements, the locating principle to be used and the exact contact types and positions for locating, supporting and clamping have to be determined in the fixture planning phase. Considering the datum surfaces defined in the setup plan, a set of locating points must be identified that allows for a stable and reproducible positioning of the WP. Further, points/areas for clamping the WP, the clamping force direction (top or side clamping), the magnitude of the clamping forces and, if required, a clamping sequence, i.e. a sequence of engaging the clamps, need to be defined. Clamping planning needs to ensure that the WP stays accurately located during machining and that the clamping and reaction forces are equilibrated (KANG & PENG 2008, p. 425). If necessary, supports are planned in addition to the locating points/areas to stabilize the WP and reduce WP deformation during clamping and machining. The selection of all contact points/areas can be subject to further feasibility criteria defined by the requirements, i.e. the satisfaction of the collision prevention requirements. Fixture planning, thus, results in a fixture layout, a set of fixture-WP contact types and positions, clamping forces and directions and can further include a clamping sequence as well as a set of design requirements that the layout and an according embodiment must satisfy.

In the fixture configuration design phase, sometimes also called unit design phase, possible fixture embodiments that meet the previously generated fixture plan are created. This includes selecting a fixture type to be used, setting certain design variables of a reconfigurable fixture or selecting fixture components and determining their position and orientation to build a modular fixture. The phase can be divided into conceptual and detailed design. The result of this stage are one or more FD embodiment candidates, often just called FD candidates, for the given workholding job.

In the last stage, fixture design verification, the generated fixture plans and fixture configuration designs are verified for feasibility. A feasible fixture plan and configuration design has to satisfy the requirements. Diverse analysis methods are possible or necessary for this verification. Among the most common methods are structural stability analysis, interference analysis and fixture accessibility analysis. The analysis results cannot only be used to verify a plan or design against a requirement but also to evaluate and rank multiple design candidates for a given fixture problem to determine the best alternative. The results of the verification phase can be manifold, depending on the analysis methods used, but the main goal is to return a feasible or optimal FD that can be used for production of the given part.

## 2.2 Computer-Aided Fixture Design

All FD phases and the tasks involved are knowledge and labor intensive and often interact and depend on each other. This can make FD a very iterative and time-consuming process. According to RONG et al. (2005) fixture designers often require ten or more years of experience to generate good designs, and trial-and-error is still a common method in industry for fixture design verification. With the rise of flexible manufacturing in the 1980s the need for a more efficient FD process emerged. This led to the development of computer-aided fixture design systems (CAFD) (BOYLE et al. 2011). These systems are either embedded in commercial CAD systems or are implemented as stand-alone applications. CAFD systems can help a user in generating and verifying a FD for dedicated or flexible[1] fixtures. Different approaches for computer-aided setup and fixture planning, the determination of fixture configurations and analysis-based verification exist. CAFD systems can be interactive, semi-automatic or automatic. The majority of systems that have been developed in academia and industry are interactive or, more recently, semi-automatic, with interactive systems having the drawback that they still require user expertise and manual effort. In comparison, only very little research has been carried out on fully automatic fixture design (AFD) systems and no commercial AFD systems are known to the author.

CAFD and the tasks in workholding are strongly related to robotic grasping and much of the research work done in the two fields covers similar ground. However, as NEE & TAO (2004, p. 57) points out, there are some fundamental differences, the most important being that in robotic grasping normally all contact elements (the fingers) actively exert a force and that grasping is less subject to locating accuracy requirements. Therefore, only the field of workholding is considered for the state-of-the-art review presented in the following. The following sections focus on common approaches and methods used in CAFD for the different FD phases presented in the last two decades. Several survey papers on CAFD research exist that present further approaches and older findings (TRAPPEY & LIU 1990; HARGROVE & KUSIAK 1994; BI & ZHANG 2001; CECIL 2001; PEHLIVAN & SUMMERS 2008; WANG et al. 2010; BOYLE et al. 2011). Also, setup planning is not further reviewed because it is excluded from this work, as mentioned in Section 1.5.

### 2.2.1 Fixture Planning

Fixture planning has been the main focus of the CAFD research community. The main tasks in this phase are the definition of FD requirements and the determination of a fixture layout plan (comprised of a locating, clamping and supporting plan) that satisfies the requirements. Generally, this means finding a set of contact entities (points, lines or surfaces) at the WP that can be used for locating, clamping and supporting. As the two phases are strongly intertwined, fixture layout planning is sometimes connected with setup planning and carried out iteratively (JONEJA & CHANG 1999). Often fixture planning is also connected to verification in order to check the fixture plan against the requirements and drive the generation of a feasible or optimal plan based on the verification results.

[1] Self-adapting flexible fixtures, such as e.g. phase-changing fixtures, do not require fixture design beyond the point of setup planning while especially modular and reconfigurable fixtures require design efforts.

### Fixture requirement representation

Although important for the FD process, a comprehensive representation of fixture requirements is only presented in a very limited number of research papers. HUNTER et al. (2006) base their FD approach on a functional requirement definition with a focus on physical and constraining requirements. An information model that includes physical and affordability (FD cost and time) requirements is presented by MERVYN et al. (2006). BOYLE et al. (2006) use and present a taxonomy of FD requirements that structures requirements into six classes. In a majority of the reviewed work, however, requirements were not explicitly modeled or addressed but mainly used for design verification.

### Fixture layout planning

Much more research work has been reported in the area of fixture layout planning including locating, clamping and supporting planning, with a focus on determining feasible contact points. The determination can be based on the creation and analysis of possible candidate points or on the alteration and verification of a given initial set of points. Among the common methods used for fixture layout planning are geometric or kinematic analysis to find points that restrict WP movements by form or force closure and rule- or case-based reasoning, where plans are created using rules based on formalized FD knowledge or a comparison to previous and similar fixture problems.

An example for a geometric analytical method is given in (BROST & GOLDBERG 1994): an algorithm that generates and ranks sets of locating and clamping points for a grid-hole modular fixture. The points are selected such that they achieve form closure for 2D polygonal parts. The locating plans are generated by determining all possible combinations of the WP position (through translation and rotation) and the position of three horizontal locators and a clamp, such that a set of WP edges matches the hole-grid pattern. A quality metric, e.g. considering collisions or locating accuracy, can then be used to filter the candidates. The approach was later re-used and extended to 3D parts with not only planar but also cylindrical surfaces (WU et al. 1998a; WU et al. 1998b). However, only a framework with algorithms was presented but no implementation.

Among the first to apply kinematic analysis for fixture planning was ASADA (1985) who used a Jacobian matrix of contact normal vectors with a set of conditions for total WP movement constraint and loading/unloading accessibility to check a set of given contact points. A generation of the points or re-planning in case of infeasibility was not discussed nor an implementation of the approach presented. BAUSCH & YOUCEF-TOUMI (1990) also use kinematic analysis to evaluate a given fixture plan (contact points) with regard to WP restraint. The analysis is based on screw theory, which reduces any rigid body motion with six DoF (three translational and three rotational) into a translation along a single axis and a rotation around that axis denoted as a screw vector. This drastically reduces the complexity of the analysis. ZHENG & CHEW (2009) presents an algorithm that finds a minimal subset of contact points in 3D from a set of discrete candidate points based on 1) force-closure using wrenches[2], a force

[2] Wrench theory reduces several forces and moments applied to a body to a single translational and rotational force around a screw.

formulation based on screw theory, and 2) re-planning to maximize the distance between points in order to increase the locating stability. The candidate points must initially be generated by some sort of WP contact surface discretization.

Rule-based systems use a set of heuristic rules, often in the style of *If-Then* statements, to model layout design knowledge. In most systems a routine of questions is presented to a user in an interactive or semi-automatic system and a fixture plan is created based on the answers. An inference engine and rule-base is used to reason about the rule selection and application. NEE & TAO (2004) presents a system that uses rules for the interactive selection of locating, clamping and supporting surfaces, which are then discretized into candidate points using a fixed interval method. In a more advanced rule-based system fixturing principles are modeled as rules, which allow for setup planning and fixture layout design in a semi-automated system (JONEJA & CHANG 1999). Here, a user has to select the locating surfaces in a 3D CAD environment and locating as well as clamping points are chosen and verified for WP form- and force-closure automatically. In the same vein, the systems presented in (SENTHIL KUMAR et al. 2000) and (HOU & TRAPPEY 2001) can generate candidate locating points and select a set of points from the candidates. The system presented in (PENG et al. 2011) uses rules to select a locating scheme and a combination of rules and fuzzy-logic to determine locating and clamping surfaces. Another system uses screw and wrench theory to check the equilibrium of machining, clamping and reaction forces to evaluate WP stability of a given fixture plan and then uses rule-based reasoning to drive the re-planning or re-design of the contact points based on the evaluation results (ROY & LIAO 1999; ROY & LIAO 2002). The critical points in rule-based systems are the development of a rule-set that models the knowledge to the required degree and allows for reasoning. The lacking expressivity of the rule languages often makes it hard to model complicated contexts.

Case-based reasoning systems use a database of previous fixture problems and solutions together with methods for indexing cases and measuring similarities between cases based on the index. When given a new fixture design problem a case-based reasoning system indexes the case by some criteria, such as WP geometry or tolerances, and then tries to determine one or more similar, previous cases in the database that can be reused or adapted to the actual problem. An example of a case-based reasoning system for modular fixture planning is presented in (LI et al. 2002). A database of fixture components and assembly relations is developed and the indexing is based on functional decomposition of the components. RONG et al. (2005) and BOYLE et al. (2006) also use case-based reasoning for the design of modular fixtures. Their system classifies fixture designs in two libraries, one for conceptual and one for detailed design information with fixture planning being addressed in the conceptual design. As WANG et al. (2010) state, the mentioned systems use one automatic case-based reasoning cycle to come up with a fixture layout plan that often does not perform well and is far from optimal. To overcome this shortcoming and generate better layouts, he proposes an interactive system that uses multiple case-based reasoning cycles (WANG & RONG 2008). After each cycle, the system presents a selection of similar cases from which the user has to pick the best case. The critical points in case-based systems are the criteria used for indexation and similarity comparison and the automation of these tasks as well as the creation of an initial database of cases.

### Fixture layout optimization

Fixture layout planning often includes optimization methods to find optimal contact points or clamping forces. The general idea is to generate or start with a layout candidate and then going into a loop of simulation, evaluation and modification, with the objective of improving WP stability or machining accuracy. The finite element (FE) method is the most prominent analysis method used to simulate the behavior of the WP under machining and clamping loads. The method is based on the discretization of a model into small finite elements and allows for the calculation of the displacements of these elements caused by loads and subject to boundary conditions. Using the FE method, WP stability can be evaluated by calculating e.g. the reaction forces at locating or clamping points while locating accuracy can be evaluated via the WP or fixture component deformation. However, other optimality criteria may also be used (ZHU 2009). For the FE analysis, the effects of the fixture-WP contacts are modeled as boundary conditions such as displacement constraints (rigid fixture) or spring-elements (elastic fixture) at certain nodal positions. The loads that result from the machining and clamping forces are calculated and applied to the WP. The optimization then changes the positions of the boundary conditions with the objective of minimizing the deformation or reaction forces.

Based on this approach, the algorithm presented in (DE METER 1998) tries to optimize the position of three vertical WP locating points with the objective of minimizing the maximum WP displacement caused by the machining and clamping loads. The clamping loads are always applied directly above the locating points, which are modeled as rigid points. The algorithm requires a starting set of points. KASHYAP & DEVRIES (1999) present a similar approach that also takes tool locations, i.e. multiple machining force application points, and kinematic closure into account. LI & MELKOTE (2001) present a way to optimize the clamping forces for multiple clamps with the goal of minimizing WP deformation. The machining forces are applied as a set of static loads and elastic contact between the WP and the fixture points is assumed. An approach to minimize the necessary clamping forces by finding an optimal position of the clamping points subject to WP stability determined via kinematic analysis is shown in (MARIN & FERREIRA 2002). A more detailed FE model, using spring-gap elements to model elastic contacts and considering drilling and milling forces as line or area loads and clamping forces as point loads is developed by AMARAL et al. (2005). The model is used in an optimization of the positions of locators and clamps and the magnitude of clamping force with the goal of minimizing the WP deformation.

The approaches mentioned above use custom optimization methods. A more generic method that has been widely applied in FD optimization are genetic algorithms. KRISHNAKUMAR & MELKOTE (2000) combine FE analysis and a genetic algorithm to minimize WP deformation by finding optimal positions for three horizontal locators and two clamps. The fixture problem is, hence, reduced to 2D and an initial fixture plan is given. The clamping forces are included and the machining forces are modeled over the complete tool-path. KAYA (2006) extends the work by not limiting the possible positions to nodal points of the FE mesh and by keeping track of previously analyzed designs in the genetic algorithm. This reduces the number of function evaluations drastically and makes the approach much faster. CHEN et al. (2008) uses an initial fixture plan and candidate contact regions with a genetic algorithm for a multi-objective optimization. The presented method can optimize contact points and clamping forces and the

FE model uses coulomb friction at the contact points and a quasi-dynamic machining load modeled by applying peak machining forces for every feature sequentially.

All of the presented FE-based approaches make some simplifying assumptions in the FE model, e.g. by treating the fixture or the fixture components as rigid, neglecting frictional effects or regarding only point contacts. Further, the machining force calculation is not explained and the load application is simplified by reducing the forces to a single or multiple static loads. In reality, the machining forces are dynamic with changing values and directions.

Most fixture layout planning approaches described in literature reduce the problem to 2D by projecting the WP boundary onto the primary locating plane (x-y plane). If the third dimension is considered, the height or z-value of the points is added in a later step, e.g. by using half of the contact surface height. Further, the approaches are only built and evaluated for modular machining fixtures, only determine contact points (no lines or surfaces) and are not linked to fixture configuration design.

### 2.2.2 Fixture Configuration Design

Fixture configuration design, also called fixture design synthesis, deals with the generation of fixture embodiments that meet the previously generated fixture plan. Although many different approaches or methods have been described in literature, most use some sort of fixture component library and reasoning or analysis to determine a set of fixture components from that library as well as an assembly plan to meet the geometric details of the fixture plan. Among the common approaches are again rule-based reasoning, case-based reasoning and analytical methods.

TRAPPEY & LIU (1993) use discrete search and heuristic rules to determine a high-level conceptual fixture design based on the 2D projection of an arbitrarily shaped WP. The method can identify whether vertical or horizontal locators and whether top or side clamps should be used and it can determine their locations on a baseplate. However, an actual selection of fixture components or a detailed design of a fixture assembly is not addressed. In (SENTHIL KUMAR et al. 1999) the combination of a genetic algorithm and a neural network is presented. The genetic algorithm generates configuration design candidates which are evaluated using the neural network. Much like in the work by TRAPPEY & LIU (1993), the "best" type of components for locating and clamping can be determined this way, but no fixture plan and no detailed configuration design are created. The system by NEE & TAO (2004) uses geometric-reasoning and a fixture component library to select modular fixture components according to the previously generated fixture plan. The system semi-automatically generates a 3D assembly model of the fixture.

JONEJA & CHANG (1999) presents a tree-style data model to represent fixture assemblies via the components used and their physical mating condition. Heuristic rules are applied to determine fixture components and configurations that meet the previously generated fixture layout plan. The approach includes the rule-based selection of a basic fixture type, either a vise-type or a modular fixture. A case-based reasoning method for conceptual fixture configuration design is presented in (BOYLE et al. 2006) and (WANG & RONG 2008). Based on the similarity of the fixture layout plan a solution to a previous fixture configuration design problem is retrieved

from a database. For detailed design, the solution still needs to be adapted to the actual case. PENG et al. 2011 uses case-based reasoning in a similar way.

A fixture configuration design system for modular fixtures based on an assembly graph is presented in (RONG & BAI 1997; BAI & RONG 1998). Algorithms search for all suitable fixturing components based on the graph and generate an assembly of the components with correct positions on a base-plate. The assembly generation is subject to an interference check between the components. An implementation in a CAD-environment allows for the 3D representation of the designs. In (WU et al. 1998a) an extension to check the design against further constraints is presented.

Through the review of fixture configuration design approaches it becomes visible that only more advanced and holistic CAFD systems include this FD phase. The result is generally a 3D assembly model or an assembly tree or graph of the fixture which can then be used for manual assembly. As in fixture planning, a verification of the results against certain design requirements is often included and used for the evaluation of design alternatives. With the exception of the system presented by JONEJA & CHANG (1999), all approaches mentioned only consider the generation of modular fixture designs. The configuration design approach for a pin-array type vise presented by WALLACK & CANNY (1996) is a further exception.

The approaches and systems that are capable of fixture configuration design only use existing fixture components. A topic that has not yet been covered in literature so far is the computer-aided or automatic generation of new fixture components or fixture component models. The research field of computational generative design covers the fully- or semi-automatic generation of design candidates. As it is important for the use of the flexible vise, this topic is discussed in Section 2.3.

### 2.2.3 Fixture Design Verification

The fixture plans and configuration designs generated in the previous FD process phases need to be verified to ensure that they satisfy the design requirements, to evaluate and compare the performance of design candidates or to allow for the optimization of the plan or design. This is done in the last FD process phase, the fixture design verification or fixture analysis phase. As already mentioned, this phase is often carried out iteratively with the previous phases in a *generate-verify* loop.

Some of the common constraints that a fixture design must satisfy to be considered “feasible” are geometric, accessibility, force and deformation constraints. Geometric constraints reflect that a fixture must be able to accurately locate a WP with respect to machining accuracy and tolerance requirements. To meet accessibility constraints a fixture must be designed such that it is easy to (un-)load the WP and such that there are no interferences between the fixture and the machining tools, between the fixture and the WP or among the fixture components themselves. Force constraints represent the need of the fixture design to resist clamping and machining forces and that the clamping force ensures WP stability. Finally, to satisfy deformation constraints the stiffness of a fixture must be sufficient to keep the WP deformation and displacement within the limits allowed by the tolerances (KANG & PENG 2008). Many different analytical approaches to verify design candidates against the constraints are presented

in literature and some of the key methods have already been mentioned. The main fields of interest in CAFD verification research are stability analysis, tolerance analysis, deformation analysis and interference analysis.

Stability analysis is used to verify that a WP is accurately restricted against any movement caused by clamping and machining forces. This way, a design can be checked for geometric and force constraints. The most commonly applied approaches are screw theory to check locating completeness, i.e. form closure, and wrench theory to verify the equilibrium of the external forces and the internal reaction forces caused by the locators, i.e. force closure (BAUSCH & YOUCEF-TOUMI 1990; JONEJA & CHANG 1999; ROY & LIAO 1999; ROY & LIAO 2002; MARIN & FERREIRA 2002; KANG et al. 2003d; ZHENG & CHEW 2009) .

Tolerance analysis is used to check geometric and deformation constraint satisfaction. WANG (2002) relates localization errors to the geometric errors of features to be machined in order to derive an actual position of the features that can then be used to update the machining code to compensate the error. The approach, therefore, detects the locating error that the FD causes. Other approaches use the inverse process, i.e. determining a locating error range from the known WP tolerances and ensuring that a FD is within this limit. KANG et al. (KANG et al. 2003b; KANG et al. 2003c) directly relate the deviations of WP features to locating tolerance errors to find a FD that is within the tolerance limits. Building on this, an approach to verify if locator deformations caused by external forces can satisfy given part tolerance values is presented in BOYLE et al. (2006).

### Deformation analysis using FEM

Deformation analysis, mainly carried out using the FE method, is used for geometric, force and deformation constraint checking and is one of the most commonly used analysis methods in CAFD verification. Multiple FEA-based approaches to optimize support and clamp locations and sometimes also clamping forces with the objective of minimizing the WP deformation have been presented in literature. They are generally based on simplifying assumptions like linear elastic workpieces, rigid fixture bodies, frictionless point contacts and static machining loads applied to a single or several substitutional positions (DE METER 1998; KASHYAP & DEVRIES 1999).

To model the fixture contacts more realistically, spring elements with friction can be used, as presented by AMARAL et al. (2005), who utilize linear spring-gap elements to model locator stiffness. RATCHEV et al. (2007) use empirically derived non-linear spring rigidity profiles to accurately represent the stiffness effects of different fixture components and includes the effect of clamping forces as pre-loads on the springs. Springs and Coulomb friction are used to model the point contacts in (CHEN et al. 2008). Further, they include a more detailed representation of the machining loads. The peak force caused by the machining of each feature is calculated and applied as a series of element surface loads according to the cutter tool path. A study that concentrates on the geometric details of the contacts between fixture components and WP surfaces is presented by ASANTE (2008). Realistic surface-surface contacts are used instead of the commonly used (idealized) point-surface contacts. The aim is to check whether the applied clamping forces can cause contact (or reaction) forces that withstand the machining forces, taking tangential forces at the contacts caused by friction and the pressure distribution at the

contacts into account. An experimental study on the effect of the contact stiffness of fixture components on the analysis results is presented in (ZHENG et al. 2005; ZHENG 2005). SIEBENALER & MELKOTE (2006) explore the effects of finite element model parameters on the accuracy of WP deformation. The modelling of contacts as boundary conditions is compared to the actual inclusion of the fixture component models and the inclusion of the complete fixture model in the FE geometric model. Further, the sensitivity of the results to changes in the friction coefficients, the effect of spherical-planar or planar-planar contacts, which represent spherical or planar locators, and the effects of different mesh densities are studied.

### Interference analysis in fixture design

Interference analysis can be used to check the satisfaction of certain accessibility constraints. In an assembled machining fixture including the WP to be held, there should be no interferences between the WP and the fixture, apart from the wanted fixture contacts, between the fixture components themselves, apart from the dedicated contacts used for assembling the fixture components, and between the cutting tool and the fixture.

The detection of interferences between the WP and the fixture can be done using static, geometry-based analysis methods. In general, the methods find Boolean intersections between models of the WP and the fixture, using 3D solids or 2D projections. This problem is considered solved and such interference analysis methods are commonly integrated in most modern CAD systems. Thus, no work of this kind has been reported in recent CAFD research.

Interferences between fixture components in modular fixture design have been a subject of investigation in (DAI et al. 1997), (JONEJA & CHANG 1999) and (PENG et al. 2010). These approaches use a component library, a component data model that defines the contact or assembly features of each component and a tree, hierarchy or graph to represent the assembly structure. In this representation scheme the allowed mating conditions between components are modeled. By only allowing for assemblies that satisfy this condition or by checking a final design for the satisfaction of this condition with geometric reasoning, interferences between the components can be prevented or detected.

The most difficult interference analysis is the detection of collisions between the WP and the cutting tools (BOYLE et al. 2011), as the movement and possible changes in orientation of the cutting tools during the machining process make it a dynamic problem. Different approaches to detect dynamic collisions of solid bodies have been developed in different research areas, such as robotic path planning, computer-aided manufacturing or computer graphics. In (JIMÉNEZ et al. 2001) an overview of collision-detection approaches is given. According to that, several approaches exist that solve the problem as simplified dynamic analysis, e.g. using bounding boxes instead of the real (complex) shape of the moving object, or by transferring the dynamic problem into a static one, e.g. by creating a swept volume of the part in movement using the trajectory. An important factor in interference analysis is the accuracy or resolution of the detection. For FD verification, a high accuracy is required, as even minimal interferences would lead to an unwanted alteration or even destruction of the fixture. Further, different tool and fixture shapes need to be processable.

An approach that meets these premises is presented by HO et al. (2001). A dynamic interference analysis between a cutting tool and an environmental object, such as a WP or a fixture, based

on a CSG representation of the tool and a point-cloud representation of the environmental object is illustrated. The focus of the work is on the ability of real-time analysis, which makes the approach applicable, e.g. for force control of haptic devices. As real-time capability is not an issue in tool-fixture interference analysis, no application of the approach in this area is known. An interference analysis especially for FD verification using a cutter swept volume and a static 3D interference check with the fixture volume has been presented in (SENTHIL KUMAR et al. 2000; KOW et al. 2000). The volume that the cutting tool requires during the machining process is created by sweeping the 2D cross-sectional area of the tool along the tool-path. This cutter swept volume is then intersected with the fixture volume to detect collisions. The tool-path information is stored in and extracted from a native, system-internal file of the commercial CAD system used. The system can only sweep cylindrical tool shapes as the 2D area to be swept is created using the tool center point and the height and radius information of the cylinder. More complicated shapes, as for example caused by a tool holder, cannot be processed. HU & RONG (2000) present an approach for tool-fixture interference analysis that is supposed to be less computationally intense than swept volume approaches and, thus, more efficient. Instead of using 3D volumes the 2D contours of modular fixture components plus the height information is used. These contours are expanded by the tool radius. This way, the tools can be represented as a dot for 3-axis machining or as a line segment for 5-axis machining and the tool-path can be represented by moving the dot or the line segment along a 2D line. Interferences are detected by intersecting the expanded 2D contours with the dot or the line segment. Again, this approach only works for cylindrical tools and additionally requires that the fixture components are dissected into blocks or cylinders.

From the presented review of research efforts in CAFD it can be seen that at least one sort of verification is included in a majority of CAFD approaches, be their focus on fixture planning, fixture configuration design or both. However, a FD should satisfy all the core constraints mentioned at the beginning of this section to operate securely in an industrial setting. As the constraints are widely different, several types of analysis are necessary to cover them all. In literature, only some systems that include several verification aspects are presented, like the one presented by KANG et al.. In a series of papers they introduce a framework for CAFD verification that structures different approaches into geometric constraint, tolerance, stability, stiffness and accessibility analysis. Further, they present a system that provides analysis for locating accuracy and completeness (i.e. geometric constraint), WP stability and compliance with tolerance requirements, all based on geometric and kinematic models (KANG 2001; KANG et al. 2003a; KANG et al. 2003c; KANG et al. 2003d; KANG et al. 2003b).

### 2.2.4 Automated Fixture Design and Fixture Configuration

Besides the term CAFD, the expression automated fixture design (AFD) systems is sometimes used in literature. Despite the name, the systems are usually interactive or semi-automatic but not fully automatic. The term is, therefore, rather used to express that all phases of FD are covered within the approach. Some so called AFD systems do not even cover all phases, like the work of WILLY et al. (1995) that does not include fixture configuration design.

Some examples of AFD systems that include all phases are the algorithm for complete FD using a limited set of modular components presented in (BROST & PETERS 1998). The system includes

all FD phases but requires several inputs besides the WP and the fixture component models, such as geometric access constraints modeled as volumes in the WP model, quality metrics, Boolean flags to indicate design choices, such as top or side clamping, and further data. A semi-automatic system that requires less input and uses a knowledge-base, several verification methods, a component library and information data model is presented by HOU & TRAPPEY (2001). The system by KUMAR et al. (SENTHIL KUMAR et al. 2000; KOW et al. 2000; SENTHIL KUMAR et al. 1999; DAI et al. 1997) works semi-automatically and includes a variety of verification methods, including tool-fixture interference analysis. The system by RONG et al. (RONG et al. 2005; BOYLE et al. 2006; MA et al. 1999; WU et al. 1998b) uses case-based reasoning in combination with functional requirement decomposition and a series of analytical methods. The user needs to provide a 3D WP model and a file containing the machining operation data and define some design preferences in the style of utility curves to drive the case retrieval. The FD generation and verification is then done in a semi-automatic manner. The system by NEE & TAO (2004) requires the interactive definition of a setup by selecting WP surfaces for machining in a CAD environment. After that the fixture planning, configuration design and verification, again including tool-fixture interference analysis, are done semi-automatically mainly based on heuristic rules and geometric-reasoning.

All of these approaches generate 3D models of modular fixture assemblies, in a CAD-like environment, that are verified to a certain extent. However, no fully automatic FD system exists so far, since full automation of the complete FD process is thought to be too complex, too knowledge intensive and too intertwined with the design-to-fabrication process to work. Based on the 3D assembly models and, if created, the hierarchical or graph-like assembly plans, the fixture has to be manually assembled. Although it would greatly increase the degree of automation of manufacturing systems, the aspect of linking CAFD and the actual physical assembly or reconfiguration of flexible fixture devices themselves has found little interest in research. Most examples that have been presented use robots to build the fixture device from modular components based on the assembly plans generated with the respective CAFD system.

Such a system is presented in (GIUSTI et al. 1991) for the embodiment of assembly fixtures from modular fixture components that are especially designed to facilitate the automatic handling and assembly. This is mainly realized by using a grid-hole pattern base plate with tapped holes. The CAFD system can design the fixture and plan the assembly but the robot has to be programmed off-line. Further, the coarse grid-pattern causes the need for manual adjustments of fixture components after robot placement. SHIRINZADEH presents a system that combines CAFD with the robotic assembly of a modular fixture, built on an electromagnetic base-plate, for a computer-integrated assembly station. The assembly robot that is already present in the system is used for the fixture assembly task and the programs that drive the robot can be directly generated and transferred to the robot control system by a module of the CAFD system (SHIRINZADEH 1993; SHIRINZADEH & TIE 1995). A similar system for building machining fixtures is presented by ETSCHEIDT (1997).

### 2.2.5 Fixture Design Knowledge Representation

Besides solving or aiding in solving the tasks of the previously presented FD phases, the second core aspect and the basis of a (C)AFD system is the formal representation of FD data and

knowledge (WANG et al. 2010, p. 5). Most existing systems are based on 2D or 3D CAD representations. However, as geometric data is only part of the information required for all tasks and applications involved in CAFD systems, CAD representations alone are found to not be sufficient. Beyond the geometric data, an integrated, formal FD representation is needed, which is able to model e.g. WP tolerance information, material properties, tool paths, clamping and machining forces, analysis results and fixture assembly and configuration information (PEHLIVAN & SUMMERS 2008). Although the need for integrated, formal information models or knowledge representations has been identified early in CAFD research, explicit work in the field has just picked up in the last decade.

### Rule- and knowledge-based systems

A basic approach to model fixture component knowledge is to classify and structure the components based on criteria such as fixture function (locator, clamp, …), fixture mechanism (mechanical, pneumatic, hydraulic) or geometric variations. The classes can be extended with data such as material properties, tolerances, price or drawings of the components. Such an extended classification allows for the inheritance of information from parent to child classes, which can help to manage fixture component data more efficiently. However, more complicated relations, e.g. between certain data properties and their effect on the design of a fixture, cannot be modeled this way. By combining an extended classification with a rule-based system, inferences of this kind are partially possible, as presented by HOU & TRAPPEY (2001) for automated fixture component selection. JONEJA & CHANG (1999) present a hierarchical classification of fixture components that focusses on geometric data, fixture functions and the mating features of the components. In combination with a set of rules, this representation scheme is used as the basis for automated fixture configuration design in the form of assembly plan generation for modular fixtures. Another system with the same approach and purpose, but using a graph-like classification scheme, is presented by RONG and BAI (RONG & BAI 1997; BAI & RONG 1998).

Besides the combination of a classification with rule-based systems, knowledge-based systems have been used for FD knowledge modeling. In rule-based systems, the domain knowledge is modeled implicitly in the rules which act on provided data, such as the aforementioned extended classifications. In knowledge-based systems, the knowledge is modeled explicitly and an inference engine is used to allow for an interaction of the user and the modeled knowledge. DAI et al. (1997) discuss the use of a knowledge-based system with a CAD environment to model fixture design knowledge and to create a modular fixture component database. The knowledge-based system uses an object-oriented language to describe relationships between the objects. This way, the fixture design knowledge, such as mating conditions between components or the components and the WP according to locating principles, can be modeled. The system is also based on a classified fixture component database. A hierarchical, tree-like data structure is used to model the actual assemblies of fixture components into units or complete fixtures via the established mates. HUNTER ALARCÓN et al. (2010) present a knowledge-based system for the design of modular fixtures built from several different models and representations. The used models are formal or semi-formal, meaning that not all knowledge is defined rigorously in a data model. The representation of a FD is split into a functional solution (HUNTER et al. 2006) and a detailed design solution, i.e. a fixture embodiment built from a repository of modular

fixture components (HUNTER et al. 2005). Another knowledge-based system for fixture design in the area of machining aeronautical parts is presented in (RÍOS et al. 2005). The work focusses on retrieving and structuring the knowledge from experts to enable automated fixture configuration design.

From these approaches it can be seen that both rule-based systems with classified databases and knowledge-based systems can be used to represent FD knowledge. These systems, however, are based on special rule- or knowledge-based engineering frameworks and developed for a specific type of fixture components. Therefore, they are neither platform nor application independent and they also do not support seamless data exchange or communication between remotely distributed design engineers or different software applications. According to CECIL (2001) and PEHLIVAN & SUMMERS (2008), however, these are essential points that modern CAFD systems need to address to allow for the integration of currently disparate approaches and applications and to push the acceptance of CAFD systems in industry.

### XML-based information models

One approach that has been adopted by researchers in the CAFD field is the use of the extensible markup language (XML). XML is a markup language developed for use on the worldwide web to encode documents in a formal, general and easy-to-use way that can be interpreted both by humans and by machines (BRAY et al. 2008). FAN et al. (2010) present three different XML-based information models that cover fixture assembly configuration data, fixture design boundary condition data, to model the input for an FE analysis, and fixture analysis loadcase results, to model the output of the FE analysis. MERVYN et al. (2006) discuss the development of four FD information models using XML with a focus on connecting FD to other design and manufacturing activities in order to support integrated design and manufacturing. The four information models cover conceptual design fixturability feedback, intermediate part fixturability feedback, fixture assembly configuration and workpiece loading instructions. Further XML models for case-based reasoning FD systems are presented in WANG & RONG (2008) and PENG et al. (2010).

These examples show that XML can be used to model FD information. A XML structure can be created or extended to include all kind of data required by different FD applications or design engineers. However, XML can rather be seen as a general exchange and serialization format as it represents syntax of data but no semantic meaning of the information. The meaning of the data stored in an XML file is generated when the document is parsed by an agent, be it human or machine. This makes the reuse of XML data models in different systems or the integration and communication of different existing systems based on the XML file hard, as the systems may ascribe disparate meanings to the same terms or different terms to the same meaning. An approach that, just like XML, also stems from semantic web technology and explicitly addresses this issue, by modeling both syntax and semantics, are formal ontologies (HITZLER et al. 2008, p. 11ff.).

### Fixture design ontologies

Formal ontologies are semantically rich and computer-interpretable knowledge representations that actively support the inter-operability and communication of software applications. In an

ontology, information is stored as categories, objects and interrelations between them to not only list terms but also express what they mean. The use of ontologies makes systems more reliable as the formal representation allows for automated consistency checking and reasoning. Further, ontologies can be completely or partially reused as shared component in different software systems (USCHOLD & GRUNINGER 1996, p. 3).

The idea of representing fixture design knowledge in an ontology has first been mentioned, but not pursued, in (HUNTER et al. 2006). AMERI & SUMMERS (2008) presents an ontology for the representation of modular fixture design knowledge for communicating, sharing and distributing fixture designs among people or agents. The formal ontology, called FIXON, is created using the semantic web language OWL DL. This language uses Description Logic as knowledge representation formalism. In FIXON some core categories and relations for modular fixture design are defined. The ontology allows for the representation of fixture layout plans following the 3-2-1 locating principle, in the form of contact points, and for a formal fixture problem definition.

Another FD ontology that serves as integrated knowledge model for a web-based CAFD system for modular fixtures is presented in FAN (2010). In analogy to his previously presented XML scheme (FAN et al. 2010), the FD knowledge is separated into several ontological representations for WPs, setup, fixture design, FEA-based fixture analysis control and a FEA-based fixture analysis solution. The knowledge representation allows for the communication of input and output data between applications for all phases of the FD process.

## 2.3 Computational Design Synthesis in Engineering

In CAFD literature, the term design synthesis is sometimes used for the act of fixture configuration design (BI & ZHANG 2001), e.g. to express the conceptual generation of a fixture assembly design. The idea of generating new fixture component models, however, has not yet been covered in FD literature so far. This is due to the fact that most reconfigurable fixture devices use existing modular components and do not allow for the machining of new components, like the flexible vise does.

Engineering design, of which fixture design is an aspect, can be seen as the act of creating new and well performing solutions for technical problems. This includes challenging creative and analytical aspects, one of which being engineering design synthesis, i.e. the generation of candidate solutions for a technical design task in the conceptual design phase (ANTONSSON & CAGAN 2001). Traditionally, engineering design synthesis is carried out manually by design experts during the design phase, often not even explicitly but implicitly as part of the design process. Making engineering design synthesis computable and, hence, automatable yields a number of advantages: Computational design synthesis (CDS) can be used to automate design tasks, rapidly generate a high number of design alternatives or reduce errors in design. Further, CDS can help to better understand solution spaces (CHAKRABARTI et al. 2011). To make engineering design synthesis computable, a formal representation of the process is necessary. Different approaches for this formalization are known.

Approaches to computational engineering design synthesis include analogy-based design, function-based synthesis and grammar-based synthesis (CHAKRABARTI et al. 2011). Grammar-

based synthesis is found to be well suited for mechanical engineering design problems because it is usually based on shape or graph representations, which are both commonly used in this domain. In grammar-based synthesis, the knowledge about synthesizing designs is encoded in formal generative grammars. These generative grammars consist of a design vocabulary and design rules that can be manually or automatically applied to an initial design to create alternative designs. Based on an analogy to natural language that is comprised of a vocabulary (set of words) and grammar rules, defining how to connect the words to create meaningful sentences, generative grammars define design languages. Generative grammars that define a shape language, i.e. one of 2D or 3D elements, can have a shape, set or graph formulation. All of these can be summarized under the term 'spatial grammars' (KRISHNAMURTI & STOUFFS 1993).

### 2.3.1 Set Grammar Formalism

The theory of shape, parametric shape and set grammars was first presented in the area of architectural design in a series of papers by STINY and GIPS (STINY & GIPS 1972; STINY 1977; STINY 1980a; STINY 1982). Shape grammars are an approach for the generation of a design (shape) through the application of rules to an initial design. The formalization of shape and set grammars are the same, but while shape grammars directly use spatial objects to represent designs (KRISHNAMURTI & STOUFFS 1993), set grammars work with symbolic objects (STINY 1982), which facilitates a computational implementation of the approach. A set grammar G consists of four elements $G \in (S,L,R,I)$:

1) $S$ is a finite set of shapes
2) $L$ is a finite set of labels
3) $R$ is a finite set of rules and
4) $I$ is a set of labeled shapes in $(S,L)+$ called the initial set

The set of labeled shapes $(S,L)+$ builds the vocabulary of the grammar.

The rules are of the form $A \rightarrow B$ and can be applied to a set of labeled shapes $C$ to produce a new set of labeled shapes denoted as $C'$. To apply a rule to the set of labeled shapes $A$, the left hand side (LHS) of the rule, has to be found in the current set of labeled shapes $C$, also called the current working shape. This process is called matching. Matching the set $A$ in $C$ can be subject to an Euclidean transformation $\tau(A)$, meaning that $A$ does not have to be present in $C$ in the exact same position and orientation. If $A$ can be found in $C$, $\tau(A)$ will be subtracted from $C$ and replaced by $\tau(B)$, meaning the right hand side (RHS) of the rule is added under the same transformation as $A$. This results in the new set of labeled shapes $C'$. The process can be denoted as shown in Equation (1).

$$C' = C - \tau(A) + \tau(B) \qquad (1)$$

As stated, a set grammar also includes a finite set of labels L. These labels can have different representations (points, dots, letters, etc.) and can have widely different uses. The scope of uses includes LHS matching simplification, adding of non-geometric information and spatial and

state constraints, the latter two being the most common uses (HOISL 2012, S.8). A spatial label can be used to define where and how a rule is applied to a working shape, while state labels define when a rule can be applied. This can be used to define the application sequence of rules. Often labels have more than one use, e.g. combining spatial and state constraints.

In Figure 2-4, the rule matching and application process and the use of labels are graphically shown on a 2D example. The rule set in this case consists of two rules, R1 and R2 as shown in the top part of the figure. The vocabulary consists of boxes and triangles. The rules make use of a label denoted as a dot. In the bottom part of Fig. 2, two examples of applying the rules are shown. The initial working shape (IWS) is the same in both cases, but the application sequence is different, leading to different results. As can be seen, the label is used as a spatial constraint in R1, resolving the symmetry of the square and exactly defining in which way the box is to be divided into two triangles. In R2 the label is used as a state constraint, making a further application of any of the rules impossible once R2 is applied. Hence, R2 can be seen as a terminating rule.

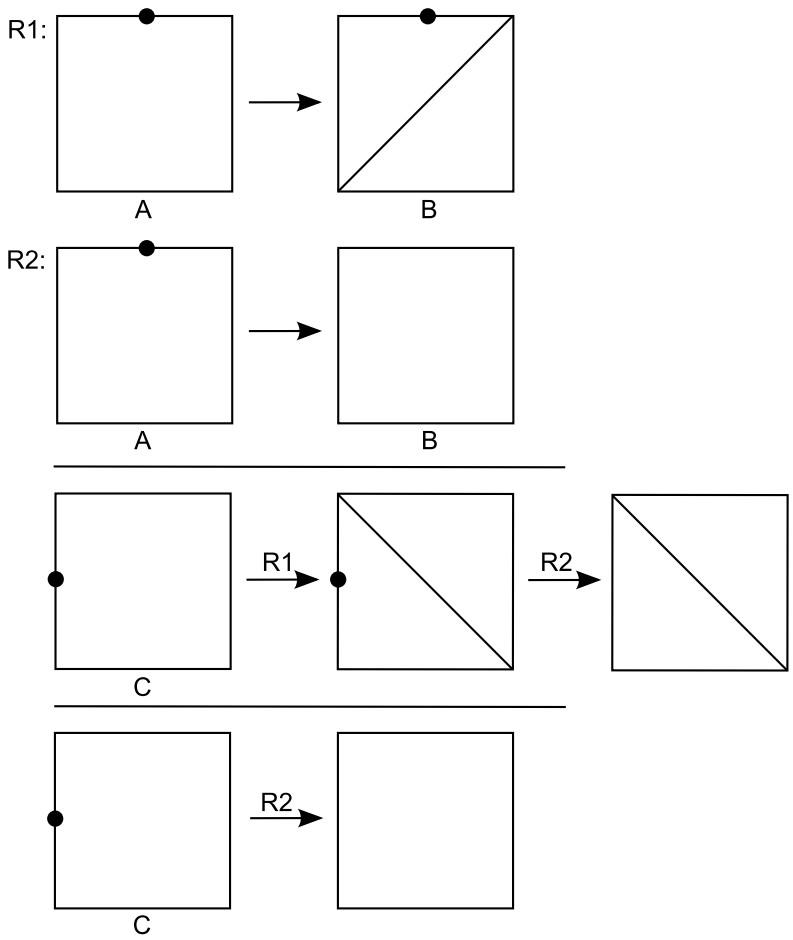



**Figure 2-4 Example of a 2D set grammar consisting of two rules and using labels (top half) and the resulting designs for different rule application sequences (bottom half) (GMEINER & SHEA 2013b)**

Making use of parameters in set grammars adds further freedom and exactness to the LHS matching process and the generation of designs by allowing for the restriction and un-restriction of parameters and the definition of parametric relations between sets of shapes (STINY 1977). As an example, the orientation, position or dimensions of the box shape A in R1 of Figure 2-4 could be unrestricted. This means that a match will be found for any type of box shape in C, no matter which size or if the shape is square or rectangular. On the other hand, a parametric relation between A and B can be set, e.g. the box on the RHS will have double the length but the same height as the matched shape.

### 2.3.2 Grammar Interpreters

In the early years of spatial grammar research, grammars were paper-based. Thus, the two stages required to generate a design with a grammar, namely grammar development and grammar application (CHASE 2002), were carried out manually or visually. Grammar development describes the definition of a vocabulary, a set of rules and an initial design to start with. The tasks in grammar application include rule selection, matching of the LHS of a rule to an object or a set of objects and the application of the rule regarding the matching conditions such as Euclidean transformations of the objects. To make full use of the approach, grammars should be implemented computationally. Among the first to publish on the aspects of implementing spatial grammars was KRISHNAMURTI. He presents an algorithm for LHS matching and rule application (KRISHNAMURTI 1981) and an implementation of this algorithm called a 'shape grammar interpreter' (SGI) (KRISHNAMURTI 1982). A review of further implementations of shape grammars can be found in GIPS (1999).

According to this review, most SGIs at the time only allowed for user interaction during the application phase, while grammar development was done hard-coded during the implementation. Further, the systems differed in which of the three application tasks, rule selection, LHS matching and the determination of matching conditions for rule application, are done by a user and which by the system. According to HOISL (2012, p. 11f.) the automation of these tasks can be used to differentiate SGI into manual, semi-automatic and automatic systems.

CHASE (2002) believed that a grammar interpreter should interactively support the tasks involved in both grammar development and application to be of help to a designer. In a recent review on spatial grammar implementations by MCKAY et al. (2012) the idea that an SGI should support a user in developing a grammar, i.e. the grammar development should be moved from a developer to the user of the grammar, was highlighted.

### 2.3.3 Examples of Applied Spatial Grammars

Many different spatial grammars have been developed since the 1970's, mainly in the areas of architecture, arts and, more lately, engineering design. Two-dimensional paper-based shape grammars, e.g. for the generation of Chinese lattice designs (STINY 1977) or ground plans for Palladian villas (STINY & MITCHELL 1978), were among the first examples published. Three-dimensional (3D) paper-based shape grammars followed, e.g. a design language for Froebel's building gifts (STINY 1980b) or for Frank Lloyd Wright style prairie houses (KONING & EIZENBERG 1981). Looking at computationally implemented spatial grammars, most of the examples found are hard-coded solutions. A system able to generate 2D cross-sections of

traditional Chinese buildings, built from straight lines, is presented in LI & KUEN (2004). In the area of engineering design, 2D grammars able to represent object shapes that can be machined with a lathe (BROWN et al. 1994) or to generate Harley-Davidson like motorcycle designs (PUGLIESE & CAGAN 2002) or front views of Buick cars (MCCORMACK et al. 2004) were presented. All of these grammars are intended for interactive design generation, meaning that the application is based on user decisions. A shape annealing approach that combines annealing optimization with a shape grammar for the generation of truss structures is presented in (SHEA & CAGAN 1997). The implementation of a 3D spatial grammar for the generation of clock gear systems from cylindrical primitives is shown in (STARLING & SHEA 2002). As the grammar combines a function and a structure grammar it is labeled a parallel grammar. In the field of electrical engineering an implemented grammar for the generation of micro-electromechanical systems (MEMS resonators) was presented (AGARWAL et al. 2000). More examples on grammars and grammar interpreter applications are given by CHAU et al. (2004), MCKAY et al. (2012) and CAGAN (2001).

These examples show the validity of using spatial grammars for conceptual engineering design tasks. As common with concept designs, the results are rather high-level, often 2D, and not production-ready. However, first attempts to use spatial grammars for detailed design have also been reported in literature lately. In the area of design-for-manufacturing, WANG (2002) present a spatial grammar implementation for the generation of 3D objects that are directly manufactured with rapid prototyping techniques, an FDM machine to be precise. As the generated objects are rather artistic sculptures, no detailed design constraints, as common in real engineering or architectural design problems, are included in the grammar. Taking it one step further, Sass presents several set grammars for the generation of architectural designs and models made by rapid-prototyping. In SASS (2006) a set grammar for the generation of housing model parts that are ready for manufacturing with 2D digital fabrication devices, such as laser cutters or routers for processing cardboard or plywood, is presented. The grammar can transfer a given 3D model into face elements with notched studs that allow for an assembly of the parts that can withstand forces caused by gravity and handling of the model. The grammar, hence, translates an early stage 3D design into constructible parts with embedded assembly features. In a similar manner, a set grammar that can translate (manually) gridded surfaces into parts with assembly features for the production with additive layered manufacturing techniques is illustrated in SASS (2008). This approach can transfer complex surfaces of any size into 3D parts that can be produced on a machine with a restricted envelope and ensure that the parts can be assembled. Both of these grammars, thus, embed functional engineering design constraints, although in a limited manner. Another grammar-based system that includes several functional constraints is the shape grammar machining planner presented by ERTELT & SHEA (2011). Here, a parametric spatial grammar is used to represent the volumes that a machining motion with a cylindrical cutting tool can remove from a workpiece. Machining planning imposes many constraints, e.g. continuity of the cutting process or applicability of certain tools. However, only a limited number of constraints, such as the possible tool motions of a 3-axis machine, are embedded in the grammar. The rest is included in the search strategies used to drive the machining plan generation.

Although the number of design constraints included in the grammars is limited, the examples still show that spatial grammars can be used for detailed design and not just for conceptual

design. This requires the transfer of functional design requirements into spatial constraints that can be embedded in the grammar rules.

## 2.4 Conclusions

CAFD and the involved tasks and approaches have been subject to research for three decades. Despite the significant contributions that have been achieved there are still some critical shortcomings.

### General shortcomings of CAFD research

From their review of recent CAFD literature, BOYLE et al. (2011) derive two main unresolved issues: the lacking focus on detailed fixture design and the segmented, non-integrated nature of current CAFD research. The first issue means that most approaches only create conceptual design representations but do not deal with the detailed physical configuration of the fixture. However, this is an important point for the automation of the FD task and to push the capabilities of CAFD systems to a level that makes them perform equally or even better than a human designer. The second issue is based on the fact that several solutions for individual tasks in CAFD exist, but hardly any integrated approaches that focus on all necessary tasks and the link between those tasks are known. This is thought to be due to the strong interdependencies of the tasks in FD, the interdependencies of FD and other tasks in the design-to-fabrication process (like CAM or CAPP) and the high number of design requirements and variables. The integrated approaches that do exist are limited in their applicability concerning fixture types, processable WP shapes and reuse in different scenarios.

Several researchers identified the lack of a commonly agreed formal FD representation as a bottleneck in CAFD tool development. Most existing representations are restricted to a certain type of fixture, mainly modular fixtures, to a certain stock of fixture components or to a specific expert system. Many methods represent fixture designs only by points in 3D space to locate, support and clamp the WP (CECIL 2001). These specific representations are not general enough to transfer the necessary information between different applications. Further, a general way of information transfer, e.g. via an integrated, application-independent and commonly interpretable information model, is required to combine CAFD approaches that are so far disparate (PEHLIVAN & SUMMERS 2008). Some advances in creating such data models using XML-based documents have been reported. However, these data models still do not represent the semantic meaning of the stored data, making reuse of the models in other systems hard if not impossible, due to the non-uniform use of terminology among different systems. It also hinders their use in collaborative or distributed environments, such as distributed development or agent-based manufacturing scenarios (AMERI & SUMMERS 2008).

The use of formal ontologies as FD knowledge representation promises to overcome these shortcomings by combining syntax and semantics in a formal, application-independent knowledge representation. Two ontologies for FD knowledge representation have been presented in literature (AMERI & SUMMERS 2008; FAN 2010). The FD representation scheme used is, however, limited to modular fixtures and found to not be general enough to represent, e.g., vise-type fixtures. The FIXON ontology focuses primarily on fixture planning and not on

fixture configuration design or design verification knowledge. The ontology by FAN includes the latter aspects but does not provide a formalization of the presented ontology, which makes a reuse, extension or refinement not possible. The ontology is developed for a specific fixture device, which makes it hard to apply it in other FD scenarios.

Considering the next step in the design process, the link between detailed fixture design and automated configuration or assembly on the hardware level, only very few systems have been presented in literature. The predominant class of systems uses either robots with dedicated grippers to assemble modular fixtures according to the created fixture configuration designs or they are based on specifically designed modular fixtures that facilitate assembly. Another type of system uses special numerically controlled flexible fixtures. All of these approaches require additional equipment and control systems, are cost-intensive and only work in specific manufacturing systems or for a limited part range.

### Device-specific shortcomings

The focus of CAFD research on modular fixtures also imposes a limitation to point or small area contacts between the fixture components and the WP. This mainly affects the approaches or models used for fixture planning and fixture verification. The approaches are, thus, not general enough to be (directly) applied to fixture types that use surface or line contacts for locating, supporting and clamping a WP, such as, e.g., vise-type fixtures.

Further, no approaches for the automatic synthesis of new fixture components have been mentioned in literature, as the use of custom-built components is not a feature that common flexible fixtures provide. CDS approaches, especially spatial grammars, have been successfully used in engineering design for the semi-automatic or automatic generation of conceptual part designs. Further, a few examples of using spatial grammars for the generation of detailed, production-ready designs that meet certain design requirements have been shown. These examples, however, were of a simple nature and did not impose many or tight design constraints as, e.g., found in FD.

In the development of a modern CAFD or AFD system, which is a goal of the presented work, these shortcomings leave a gap and, thus, opportunities for new approaches and methods, as explained in the next section.

# 3. Approach to an Integrated, Automated Fixture Design System

This section describes the approach taken in this thesis to create an integrated flexible fixture system that links automated fixture design (AFD) on the software side and reconfiguration of the flexible vise, the fixture device presented in Section 1.4, on the hardware side, to enable autonomous workholding for machining operations.

## 3.1 Research Objectives

In Section 1.5, the goal of this thesis, the determination, development and validation of a set of methods and applications that form the basis for the integrated AFD system for the flexible vise, is stated. Such a system must be capable of automatic fixture planning, fixture configuration design, fixture design (FD) verification and, thus, the generation of a FD for a workholding job with minimal user interaction based on a given 3D model of the part to be produced. All data necessary to drive the reconfiguration of the fixture device on the hardware level should be produced and stored during the FD process with the AFD system.

A design of a single-acting vise is defined by the jaws used on the anvil and the slide side, the clamping length and the clamping force. The AFD system must be able to automatically determine "values" for these fixture design variables, verify their feasibility with regard to the design requirements and allow for storing the results in an integrated information model and communicating them to the flexible manufacturing system (FMS) and fixture device control systems. The flexible vise allows for the (re-)use of existing jaws and for the machining of new jaw designs from blanks. Therefore, the AFD system needs to be able to reason about possible FD candidates built from existing fixture components and, in case no feasible designs can be found, to generate new fixture component designs that allow for the accurate locating of the given workpiece (WP). The design candidates then need to be verified to detect infeasible designs and evaluated to determine the best design among the feasible candidates. For this FD, the design variables can be stored and communicated to the fixture and FMS control systems to reconfigure the fixture device accordingly. Reusing existing jaws, that are stored in the FMS, or existing jaw designs, that are stored in a database, should be preferred over the designing and machining of new jaws for economic reasons.

Most of these tasks, such as generating fixture design candidates and verifying them, are core aspects of computer-aided (CAFD) or automated fixture design and several methods to approach the tasks have been presented in literature that must be taken into account for the development of the flexible vise AFD system. Further, the system needs to address the identified shortcomings of state-of-the-art (C)AFD systems, presented in the previous section, and the preconditions presented in Section 1.5. Taking these points into account, the focus for the development is on the following aspects:

- Creating a formal, integrated FD representation and information model that is semantically rich, platform- and device-independent, extendable and reusable and can serve as a common knowledge basis for all FD tasks and applications;
- Developing a device-independent method to automate fixture configuration design on a semantic level including the determination of useable fixture components from the database of existing components;
- Automating the synthesis of new fixture component designs that meet given design requirements;
- Verifying the generated FD candidates for satisfaction of the design requirements and evaluating them to select the best candidate for production.

Table 3-1 illustrates the focus aspects and the methods used in relation to the fixture design phases.

**Table 3-1 Foci and methods used for the different fixture design phases**

| Fixture design phase | Focus | Method |
|---|---|---|
| Setup planning | -Excluded, see Section 1- | -Excluded, see Section 1- |
| Fixture planning | Formally represent fixture designs | AFD ontology |
| Fixture configuration design | Determine possible fixture designs<br>Synthesize new fixture component designs | Ontological reasoning<br>Spatial grammar |
| Fixture design verification | Check candidates for tool-fixture collisions<br>Check candidates for WP deformation | Interference analysis<br>FE analysis |

For the first aspect, a formal AFD ontology is developed as presented in Section 4. The ontology should serve as a FD knowledge representation, as the basis for all phases of the FD process and all applications in the AFD system and as a link to the fixture reconfiguration control.

Further, the ontology must represent FD in a way that allows for the identification of possible fixture candidates among the fixture designs that can be built with the existing components by semantic reasoning based on a given workholding job. This complies with the second aspect, the fixture configuration design using existing components or component designs and is presented in Section 4.3.

A spatial grammar for the computational synthesis of new FD candidates is developed, addressing the third aspect, as described in Section 5. This method enables the flexible vise and FMS to create and use newly machined fixture components (vise jaws) for fixture configuration design. The grammar development includes the exploitation of the ability to integrate design requirements of a real engineering problem in a spatial grammar in sufficient detail.

For the automatic verification and evaluation of the FD candidates, the fourth aspect, two different analysis methods are pursued. First, the candidates are checked for interferences between the cutting tool of the milling machine and the fixture to sort out infeasible designs. The analysis is based on collision detection between the 3D model of the fixture assembly and the volume that the cutting tools require, which is determined using the machining plan of the setup. Second, the WP deformation of each remaining candidate is determined for design evaluation using the finite element analysis (FEA). The FD candidate that causes the least WP deformation is selected as the final FD that is to be used for the production of the part. This is presented in Section 6.

In the following, the structure of the AFD system and the resulting fixture design process using the mentioned approaches are described before the methods, the reason for their selection and the applied software tools or prototype implementations are discussed in more detail.

## 3.2 AFD System Structure and Process

The automatic fixture system consists of three main components: the AFD system, consisting of the applications for the FD phases, the manufacturing system, including the fixture and the FMS control systems, and a knowledge base in which the AFD ontology is the core component.

Figure 3-1 shows the link between the knowledge base, the AFD system and the manufacturing system with the focus topics of this work highlighted in dashed lines. The knowledge base contains the AFD ontology, a temporary data storage and a model repository. The AFD ontology serves as the central knowledge representation and database for all FD phases and applications. All temporary data, such as the FD candidates, their assembly models or the analysis results, are stored in the temporary database. Only if a design is found feasible, the corresponding temporary models are stored in the model repository permanently and the FD is instantiated in the ontology. The instantiation, also called populating of the ontology, is done by generating individuals and setting their data properties as well as the links to the model repository. Besides the jaw models, the model repository stores the WP models and the cutting tool models. Thus, the ontology itself does not store any CAD models but rather links to the repository.

The AFD system contains of applications for fixture planning, fixture configuration design and fixture design verification. The applications for fixture planning and fixture configuration design include the ontology editor, to run the queries and perform the reasoning, and the spatial grammar and shape grammar interpreter application (SGI), for computational design synthesis. For FD verification an application for the interference analysis and generation of assembly models from the WP, anvil and slide jaw models, as well as an application for FE analysis are included. As presented in Section 1.5, automatic setup planning is not a focus of this work. However, as it is a necessary step of FD it is included in Figure 3-1 as part of the AFD system block and carried out manually in the presented approach.

The AFD system interacts with the knowledge base in both ways. For example, the reasoning for fixture configuration design is based on the AFD ontology, the models created by the spatial grammar have to be stored in the temporary repository and the assembly model creation requires the permanently stored CAD models of the existing jaws and the WP. The knowledge

base also interacts with the manufacturing system to enable the physical fixture reconfiguration. The fixture control system, for example, requires the clamping length and clamping force values of the chosen FD that are stored in the AFD ontology. The FMS control system requires the identification numbers of the jaws and the WP to initiate their retrieval from the storage or the WP placement position for the handling robot. As the manufacturing system only requires the communication of data of feasible and stored FD a direct link between the AFD system and the manufacturing system is not necessary.

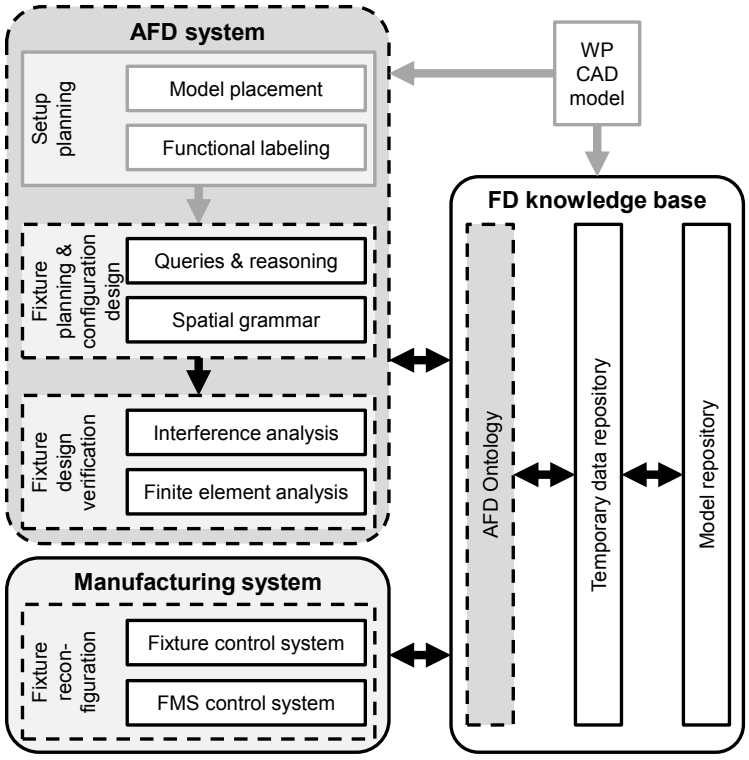



**Figure 3-1 Overall structure of integrated AFD system - link between AFD system, manufacturing system and database with focus topics highlighted in dashed lines**

The automation of the data handling, including the instantiation of WP, setup and FD data in the ontology, is outside the scope of this work and carried out manually. The instantiation can be automated using ontology application programming interfaces, existing text-to-ontology applications or a custom application for writing an ontology file. The communication and data handling between the AFD system, the manufacturing system components and the knowledge base can be automated using software agents. An agent-based approach for the CogMaSh system is presented by SCHÜTZ et al. (2011).

Using this system structure the process of creating a fixture design is structured as illustrated in Figure 3-2. The design-to-fabrication process starts with a production request and the CAD model of the part to be produced. An existing ontology-based application helps a user to pick a stock part from the parts available in the manufacturing system storage. The selection is based on the required dimensions and material properties and the application returns a ranked list of

possible stock parts for a user to choose from (SHEA et al. 2010). The selection can also be automated. Once a stock part is selected, the first step of the fixture design process, the setup planning, can be carried out. The prepared WP model is stored and used for instantiation in the AFD ontology, meaning that all necessary data of the setup, according to the ontological representation, is modeled in the ontology. This is required to allow for the next step, the reasoning over possible FD candidates.

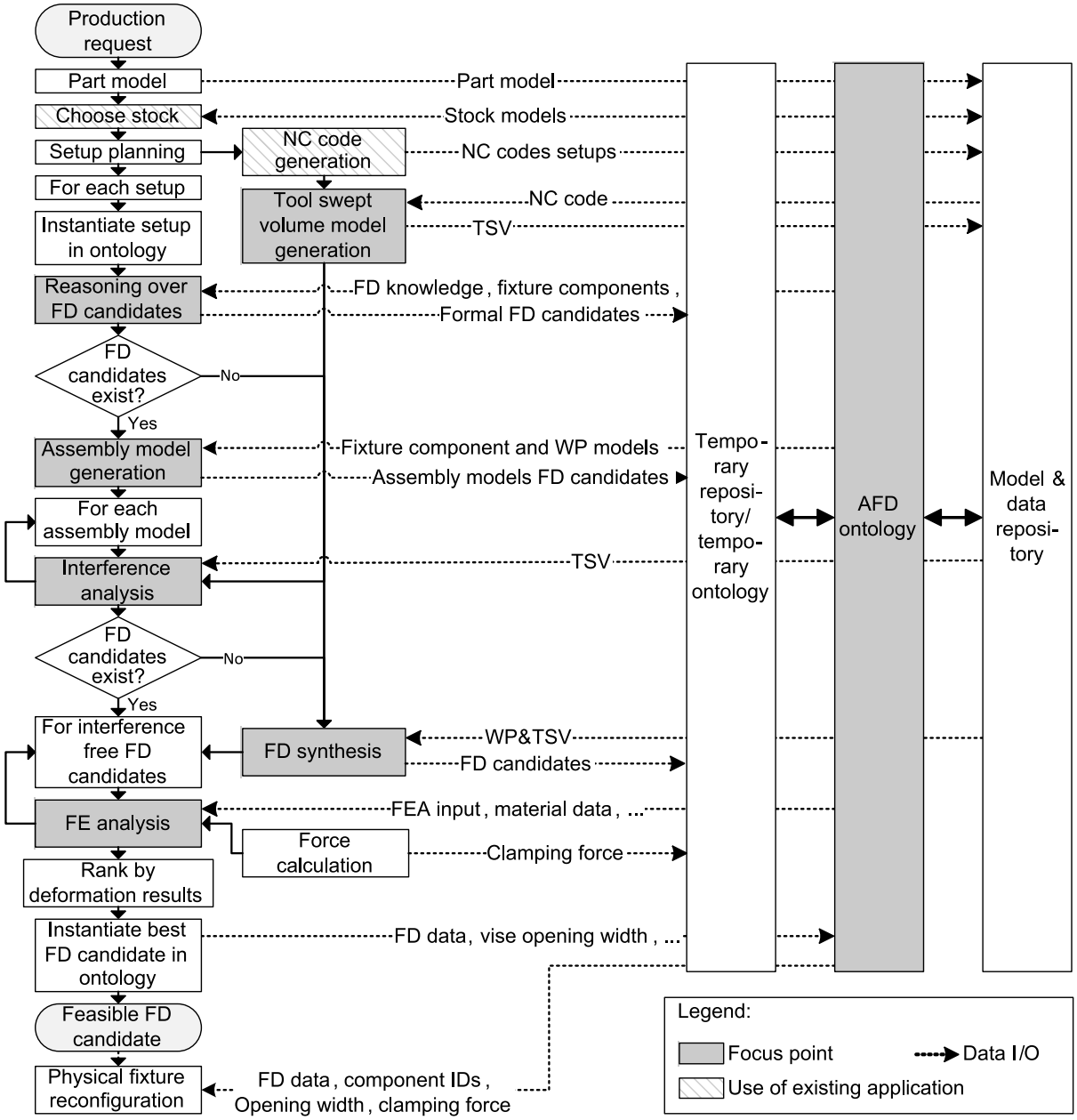



**Figure 3-2 Simplified fixture design process for the AFD system with core data transfers**

The reasoning is driven by manual queries to the ontology. The queries can be based on geometric details of the WP and setup or simply on the setup identification number. In parallel

to the reasoning, the NC code for the production of the WP is generated. In a classical design-to-fabrication process, FD is considered to happen before the manufacturing planning stage (CAM) in which the NC code is generated (RONG et al. 2005). However, in the CogMaSh concept a different structure of the design-to-fabrication process is used, in which the FD can be carried out after or as part of the manufacturing planning phase. The automation of the CAM process for NC code generation is not a focus of this work. A spatial-grammar based approach and application for machining planning, developed in the CogMaSh project, is presented by ERTELT (2012). All NC codes used to validate the approach presented here are generated using a commercial CAM application and considered given. Based on the NC code and the library of the cutting tool models of the CogMaSh milling machine, stored in the model repository, the tool-swept volume model (TSV) is generated. The TSV is then combined with the WP model. For possible FD candidates identified by the reasoning, an assembly model, containing the combined TSV/WP model and the reasoned anvil and slide jaws, is built. The assembly models are used for the interference analysis. Candidates that cause an interference are not further considered while candidates that are interference-free are passed to the FE analysis.

In case no FD candidate can be determined by reasoning or if all reasoned candidates are found to be infeasible in the interference analyses, new fixture components and corresponding FD candidates are being synthesized. By including the TSV as obstacles in the design generation process, the synthesized candidates are guaranteed to be interference-free. All interference-free candidates are then checked for the maximum WP deformation and the candidate causing the smallest maximum deformation is used for the manufacturing of the setup. The candidate is instantiated in the AFD ontology and the process is ended. In case no feasible candidate can be found, the process ends without a result. As mentioned in Section 1.5, the presented work aims at generating feasible FDs and selecting the best candidate, not at optimizing a FD.

The instantiation of the feasible FD adds all data required by the fixture- and manufacturing system control for the reconfiguration of the flexible vise to the AFD ontology. Once the WP is to be fabricated, the data can be retrieved from the ontology and communicated to the control systems. The process steps necessary for the reconfiguration on the hardware side are not depicted in Figure 3-2 but are included in the extended process scheme shown in Appendix 10.1. Further details on the hardware reconfiguration process can be found in (ERTELT et al. 2009).

The described process is setup based, meaning that FDs are determined for one setup. Often multiple setups are required for the fabrication of a part. The AFD system and process allow for that, by using the intermediate WP models that result from the machining of a setup as stock (or WP) for the subsequent setup. A simulation of the NC code can be used to create the intermediate geometric WP model with a commercial CAD system or the shape-grammar machining planning approach by ERTELT (2012). This is not a focus of this work and the intermediate models are considered given. If necessary, this model again has to be oriented and labeled to represent the subsequent setup data. This process can be repeated until the stock WP for all necessary setups determined in the setup planning stage are generated. This sums up the AFD system structure and according FD process. Next, details on the applied methods and approaches are presented.

## 3.3 Ontologies as a Formal Fixture Design Representation

As a computational knowledge representation and an integrated FD data model formal ontologies are used. In scientific literature on ontologies, the term is used with many different meanings, e.g. for glossaries, taxonomies or data models. In the presented work, the definition by GRUBER (1995) is used. Gruber states that "an ontology is an explicit specification of a conceptualization". Conceptualization means the description of the domain of discourse in unambiguously defined concepts, their relationships and attributes. Explicit specification means that all the concepts, attributes and relations are represented in a formal way. A formal representation enables software tools to make use of the ontological knowledge.

Formal ontologies allow for semantically rich knowledge modeling for a domain of interest. Further, they actively support inter-operability of applications and provide the benefits of specification, reliability and re-usability for software system development. Specification means that ontologies can assist in defining a shared understanding or terminology for one or multiple domains. The use of ontologies makes systems more reliable as the formal representation allows for automated consistency checking. Further, ontologies can be completely or partially reused as a shared component in (other) software systems (USCHOLD & GRUNINGER 1996, p. 3).

The use of ontologies as a formal knowledge representation is, moreover, especially suitable for agent communication (VAN AART et al. 2002). This is beneficial in the given scenario because the production planning system envisioned for the CogMaSh sytem and used for the control of the flexible vise is agent-based (SCHÜTZ et al. 2011). Ontologies are already used to model other parts of the CogMaSh system domain knowledge, e.g. the available machining stock (SHEA et al. 2010, pp. 258-259), and the integration or merging of ontologies is natively supported. Further, other FD ontologies exist that can be partially reused, as presented in Section 2.2.5. Ontologies are, therefore, well suited for the formal knowledge representation in the given scenario.

For the AFD system, the necessary domain knowledge is modeled in two separate ontologies: a general fixture ontology and a fixture design ontology that covers device-specific configuration knowledge and approach-specific knowledge required for fixture design synthesis, finite element analysis and interference analysis. The separation is a common process in ontology development as it facilitates the coupling or reuse of the ontologies in different systems (BORST 1997, pp. 18-24). The general fixture ontology contains entities and statements that hold for all fixture types, like a classification of different fixture components and devices. The entities are only loosely defined which simplifies the reuse of this ontology in other scenarios. The specific AFD ontology builds on the fixture ontology and adds knowledge related to FD with a focus on the flexible vise. This includes class definitions that are unique to the fixture components of the flexible vise or data property restrictions like the maximum clamping length of the vise. Further, knowledge or data that is specific for the methods or applications used in the AFD system at hand are included. This is, for example, knowledge necessary for the FD verification such as the paths to the 3D models of the fixture components.

To keep the general fixture ontology device-independent, the formal definitions that describe the domain have to be commonly agreed on by the domain community and be general enough to allow for their re-use in other scenarios or for other devices. Common informal definitions

of the domain are identified by studying literature and talking to domain experts. As it is one of the core ideas of the ontology community to reuse and extend existing ontologies when possible, the formal definitions found in the FIXON ontology (AMERI & SUMMERS 2008) and the structural ideas of the FD ontology presented by FAN (2010) are taken into account for the ontology development. The resulting AFD ontology serves as integrated information model for all involved FD applications by storing and, if queried, providing FD information and, therefore, allowing for communication among the applications. Input data of the applications can be stored as individuals and data properties in the ontology. By defining and linking the classes in the ontology using axioms and relations, the ontology becomes more than a hierarchy of concepts or a database but a knowledge model that allows for the representation of fixture designs on a semantic level. To allow for fixture planning and formal fixture configuration design, the ontology has to represent the fixture device, the fixture components, the locating principle and the setup data. Each FD is defined by a WP to be held, the setup details of the WP, the fixture components used (vise jaws), the clamping length and the clamping force.

## 3.4 Fixture Configuration Design Based on Ontological Reasoning

A core aspect of an AFD system, especially for a modular, flexible fixture device such as the flexible vise, is the automatic generation of fixture configuration designs, or FD candidates, based on the existing or available modular fixture components. All existing components and the knowledge required to configure specific FD from them is modeled in the AFD ontology. Reasoning engines, also called reasoners, in combination with respective queries are to be used to infer fixture design candidates for a given design problem based on the ontological knowledge representation. The use of such reasoning engines to infer implicit knowledge from the explicitly stated is one of the key advantages of ontologies. Thus, pursuing this approach for fixture configuration design appears logical.

The reasoner compares the setup information of the actual workholding job against the fixture candidates that can be built from the fixture components that are instantiated in the ontology, taking the axioms and relations into account that define the feasibility of a FD. As ontology reasoners are not geometric reasoning engines, the FDs and all involved entities must be represented in a way that allows for semantic reasoning over geometric configurations. This is realized by defining the functionality of the specific geometric entities of the WP and vise jaws in the ontology. For example, it is defined that a WP surface of a specific shape and locating function, e.g. a planar primary WP locating surface, must be located through the contact to a support surface of a certain jaw type. The feasibility of fixture-WP combinations are also subject to compliance with further geometric constraints like e.g. the maximum clamping length restriction. These constraints are included in the ontology via rules written in the semantic web rule language (SWRL) (HORROCKS et al. 2004). Based on the rules, the reasoner can determine all feasible fixture configurations for a given workholding job. As the feasibility criteria that can be covered in the ontology are limited, the candidates must subsequently be verified using analyses.

For certain inferences to be made by the reasoning engine the fixture components and WPs have to be instantiated as individuals with related object and data properties in the ontology. Thus, this FD configuration method only works for existing jaw designs. In case the reasoner

cannot find a fixture design candidate or the reasoned candidates are found to be infeasible in the verification process, the computational synthesis of new jaw designs becomes necessary. The development of the ontologies, the resulting formal FD representation and the fixture configuration design based on reasoning are explained in detail in Section 4.

## 3.5 Computational Synthesis of Fixture Components Using a Spatial Grammar

If possible, existing fixture components should be used for new fixture design jobs to minimize the need for new components. However, if no possible or feasible configurations using the existing components (jaws) can be found, new ones need to be synthesized and machined, to make full use of the capabilities of the flexible vise. For the computational generation of fixture component designs, a spatial grammar is developed. Spatial grammars provide a method for rule-based, generative 2D or 3D shape design (HOISL 2012) and, when using a shape grammar interpreter, can computationally generate designs with little to no user interaction. Spatial grammars can be used to explore a possible solution space and generate a high number of design candidates. Hence, they are well suited for the envisioned automated fixture design system. To allow for the subsequent analysis and graphical representation of the generated fixture component designs, a 3-dimensional approach is taken. The focus of this part of the work is on studying the applicability of spatial grammars to a real, detailed engineering design problem with several functional design requirements. This is an approach that has not yet been pursued in the area of fixture design and could be a key enabler for the complete automation of a requirement-driven FD process, which is the original motivation for this work as explained in Section 1. As a side effect, the grammar development is also a test of the applicability and expressiveness of the used shape grammar interpreter *Spapper* for such a grammar development task. For the computational synthesis of 3D designs alternative approaches, such as graph grammars or CAD-based parametric design, could be used. However, none of the possible approaches is thought to have any advantages over spatial grammars.

Thus, a 3D spatial grammar is developed that can generate jaw designs based on the WP model and setup data. The result is an assembly model of the WP and a set of newly generated jaws, i.e. one locating jaw and one clamping jaw, which serve as a FD candidate. Modeling or generating the rest of the fixture device is not necessary, as the design is not changed. To explore the possible solution space, the spatial grammar generates multiple different design candidates. Based on the WP shape, different basic shapes of the jaws and parametric variations of these basic shapes are to be generated by the grammar. By restricting the design corpus to some basic jaw set shapes it is ensured that the resulting designs can accurately locate a WP according to either the 3-2-1 or the V-block locating principle, the most common locating principles used for vise-type fixtures. As common in engineering design problems, the parameterization of the basic shapes is subject to compliance with existing design requirements. These design requirements are transformed into design constraints which are embedded in the grammar vocabulary and rules. This allows for the generation of detailed jaw set designs that satisfy the requirements and can directly be manufactured.

The grammar makes extensive use of labels. Via the placement parameters the labels can provide spatial information that is not present otherwise. For example, the position of certain

WP surfaces in space can be represented via labels. The placement parameters, label colors and label names can be used as state constraints to restrict the applicability of rules to certain states in the synthesis process. A rule can require the presence of a label in a certain color and, when applied, change that color. This hinders the rule from being applied again (HOISL & SHEA 2013). This technique can be used to control and terminate the rule application sequence and is required for the automation of the synthesis process. The label names can also be used to provide non-geometric information to a user or application, like the semantic meaning of certain geometric entities, such as "Contact Surface". Further, obstacles are used to restrict the design space. Obstacles are 3D volumes built from solids that are present in the initial working shape and cannot be touched or interfered with a solid entity generated by applying the grammar. This way, a design space envelope or a certain space that has to be kept clear can be modeled. The use of labels and obstacles is essential for the approach and the automation of the rule application as presented in Section 5.3. With a focus on design automation, the grammar must be modeled such that an automatic generation of design candidates is possible. This especially requires that the applicability of the rules is controlled, e.g. with state labels, including the termination of the grammar application process.

## 3.6 Analysis-Based Verification of Fixture Designs

To verify and evaluate the design candidates generated by reasoning or synthesis they need to be analyzed. Two essential analyses, cutting tool-fixture interference analysis and WP deformation analysis, are included in the approach. Although several other analyses are possible and may be required in an industrial setting, the two analyses included are essential for the secure operation of the fixture device and for ranking the design alternatives. The tool-fixture interference analysis is necessary to detect collisions between the cutting tool and the fixture that would lead to an unwanted alteration or damage of the fixture. A fixture design candidate that causes an interference is infeasible and not further considered. All candidates that pass the interference analysis are analyzed for the maximum WP deformation they cause given a calculated clamping force and the machining forces that occur in the setup. The deformation analysis is carried out for all feasible FD candidates to determine the one that causes the least WP deformation. As the WP deformation adds directly to the inaccuracies of machined features and, hence, to the final part inaccuracies, a minimum deflection is wanted.

### 3.6.1 Interference Analysis

To include the cutting tool-fixture interference analysis in the approach, a method and implementation for automatic collision detection based on the 3D model of the volume that the tools require for the machining of a setup is developed. The total tool swept volume model (TSV) is created by sweeping the 3D models of the cutting tools, which are stored in a model repository, along the travelling paths. Every tool used and its respective paths are extracted from the numerical control code (NC) of the WP/setup. The NC code of the setup is used to drive the machine tool and is created in the machining planning stage. As machining planning is not a focus of the presented work, the NC code is considered given. Every sweeping operation results in a 3D model of the space required by the cutting tool for the specific NC operation. The total TSV is created by merging all tool-specific volumes in one model. The TSV is then

added to the WP model to create a combined model. This is facilitated by the fact that the WP and the NC code, and therefore also the TSV, have the WP coordinate system (COS) as reference.

The combined model can be used in two ways: In case FD candidates created by reasoning are analyzed for interferences, an assembly model of the combined model and the anvil and slide jaw models, as determined by the reasoner, is created. The placement of the parts in the assembly model is based on the contact conditions between the WP and the jaws. The results are the same assembly models as created by the spatial grammar. The combined model is then intersected with the jaw models. If the result of this intersection is not equal to zero a collision between a cutting tool and the jaws exist, making the FD candidate infeasible. For the intersection, only the jaw set model is used and not a model of the whole fixture. This facilitates the automatic creation of the assembly model and the intersection operation and is considered feasible because the interference is most likely to happen with the jaws, as those are the closest fixture components to the WP and the TSV.

The second way to use the combined model is in the computational design synthesis process. The combined model serves as an initial shape that the grammar is applied to, with the TSV being defined as an obstacle. By this, it is ensured that all FD candidates generated by the spatial grammar are collision free as no parts of the generated jaws are allowed to intersect with the TSV obstacle. Using the TSV approach in combination with the spatial grammar approach, therefore, renders a subsequent collision analysis of the synthesized FDs unnecessary. This shortens the time needed for verification.

As the TSV creation and intersection approach can be used in two meaningful ways and only requires data that is already present, it is well suited for the interference analysis in the given scenario. Other common methods of collision detection, such as a dynamic intersection of the fixture components and the tool models, in an exact representation or using bounding boxes, or the use of 2D projections (JIMÉNEZ et al. 2001) may be less computationally intensive but do not create a TSV that can be used as an obstacle in the synthesis process.

As the interference analysis using this approach is computationally less intense than the structural analysis, it is carried out first to sort out infeasible FD candidates. All FD candidates that are collision-free are passed to the deformation analysis for further verification.

### 3.6.2 Workpiece Deformation Analysis

In the deformation analysis, the WP deformation under the influence of external forces and the boundary conditions that the FD candidate implies is checked. The goal is to identify the candidate that causes the least maximum WP deformation, as the deformation during the machining process directly adds to the inaccuracies of the finished part. The finite element method (FEM) is used to perform these analyses.

In the given FD scenario, the forces that act on the WP and the fixture are the gravity force, the clamping force and the machining forces. The machining forces are defined by the setup and machining plan and cannot be altered without changing the machining plan, which is predefined in the given scenario. The flexible vise allows for setting of the clamping force within a device-specific range by altering the pneumatic pressure of the slide-driving piston. The clamping force

must be as high as necessary to ensure that the WP stays accurately located during the machining process, i.e. that the WP does not move when it is loaded with the machining forces. At the same time the clamping force should be set as low as possible to minimize the WP deformation caused. The clamping force is hence computed such that it can withstand the machining forces in the most critical possible cases.

For a given setup plan, both the machining forces and the clamping forces are independent of the actual fixture design, as will be shown in Section 6.2.1. Despite this fact, the resulting WP deformations can differ depending on the FD candidate geometry as the load conditions that the forces cause depend on the geometry. To determine the WP deformation, the nodal displacements of the active surface of a FD candidate are calculated and stored in a table. The maximum absolute values for each candidate are then compared to identify the one with the lowest maximum.

The analyses are based on the geometric assembly models of the WP and the jaw set. The rest of the fixture device is not modeled to reduce the calculation effort. The effects of the fixture structure on the jaws and WP are represented in the analyses by introducing boundary conditions, such as displacement restrictions of certain surfaces or areas. As only metals, nonferrous metals and homogeneous plastics are processed in the CogMaSh system, a homogenous material model is used. The respective parameters of the materials used can be modeled in and retrieved from the material ontology that is part of the ontological knowledge base. To further facilitate the analysis, the gravity force is not considered in the approach as it has a minimal influence compared to the much higher clamping and machining forces. The machining or cutting forces are calculated for every feature machined in the setup and applied as a load on the meshed geometry. The type of load (face, edge or node load) depends on the type of feature and the machining operation. The forces are assumed to act in the most critical directions for the WP deformation, i.e. against the locating entities of the anvil jaw. In combination with the clamping force, this leads to the maximum possible deformations.

## 3.7 Software Frameworks and Data Formats Used

As the goal of this work is the creation of a basis for a fully automated flexible fixture system, all methods mentioned above need to be computationally implemented in software prototypes. To facilitate reuse by others, the implementations are solely based on open-source software frameworks. Further, standard exchange formats or file types are used where possible.

### Ontology development

The ontologies are modelled in the Web Ontology Language OWL 2 DL (HITZLER et al. 2012) using the ontology modeling editor *Protégé 4.3* (STANFORD CENTER FOR BIOMEDICAL INFORMATICS RESEARCH 2011). OWL 2 DL has a high enough expressivity to model the domain, allows for interference, using reasoning engines, and facilitates the integration with the existing ontologies, as they too were created in OWL. *Protégé* is a standard tool for modeling ontologies. It supports the use of different syntax for writing and representing class expressions and queries. In the presented work the *Manchester Syntax* (HORRIDGE & PATEL-SCHNEIDER 2009) is used for showing any formal definitions in the text as it is easier to read

and write then the classical OWL functional syntax. Further, *Protégé* supports different reasoners. In this work, the *Pellet* (SIRIN et al. 2007) reasoning engine is used, as it supports OWL2 and SWRL.

### Spatial grammar interpreter

The implementation of the jaw set grammar is done using the 3D spatial grammar interpreter *Spapper*. *Spapper* is an add-on module to *FreeCAD v0.12*, an open source CAD tool based on the *Open Cascade Technology* framework (*OCCT*). *Spapper* allows for the interactive creation of parametric rules and for manual or automatic rule application, all in a CAD-based graphical user interface (HOISL 2012). *Spapper* uses the parametric set grammar formalism and has a graphical user interface to develop rules and apply them in a CAD environment. It supports the use of a free combination of 3D solid primitives and also more complex bodies, created by Boolean operations or 2D extrusions, as vocabulary. Further, it supports Euclidean transformation, 3D labels, obstacles and the use of parameters and value ranges. In *Spapper*, 3D labels do not have a solid geometry but they have translation and rotation parameters, also called placement parameters, and can be used for different purposes. As mentioned, *Spapper* also allows for the inclusion of obstacles in the design process to restrict the feasible design space. To automate the grammar application, *Spapper* offers several options. If multiple matches for the LHS of a rule can be found within the working shape, the tool offers manual or random selection of the match. The rules can be applied both in a random or sequenced order and the number of applications and desired number of solutions can be set by the user. Hence, by setting the number of rule applications and desired solutions, the design generation can be done automatically given a set of rules that is designed accordingly (HOISL 2012, HOISL & SHEA 2011).

### Interference analysis

An implementation for the automation of the TSV approach is programmed using open-source tools. The implementation uses the *OCCT* framework version 6.6.0 (OPEN CASCADE SAS 2014a), the same framework that the spatial grammar interpreter *Spapper* and *FreeCAD* are based on, for all necessary geometric, data handling and visualization operations. *OCCT* is a framework for generating CAE applications based on a geometric kernel. Further, the *Qt* framework version 5.1.1 (QT PROJECT HOSTING 2014) is used for the implementation of a simple GUI. *Qt* is a standard for GUI implementations. The application, written in C++, can extract the tool model IDs and tool paths from the NC code. Data resulting from the extraction and the links to the cutting tool models are used together with *OCCT* functions to sweep the tool models and generate the TSV. The application can further create the assembly models, perform the intersection analysis and store the results as textual data.

### FE analysis

The focus is being put on the ability to automate the FEA using open-source tools. For the approach, the *Salome-Meca* framework is being used (OPEN CASCADE SAS 2014b). *Salome-Meca* combines the pre- and post-processor *Salome* (EDF R&D 2011) and the thermo-mechanical solver *Code Aster* (EDF 2014) with a GUI and several other modules. *Salome*

allows for the meshing of generated or imported geometry, both in an interactive GUI or using *Python* scripts. *Code Aster* allows for the script-based definition of the FEA settings, such as the boundary and loading conditions, the solver or the required result types. Using named groups of geometric entities created in *Salome* the boundary conditions can be linked to the model geometry. The actual solving process can then be carried out based on the scripts, which allows for an automation of the FEA process. For the presented approach, version 2014.1 V7_0_3 of *Salome-Meca* with *Code Aster* in version 11.5.0 is used.

### Data formats

The STEP (**ST**andard for the **E**xchange of **P**roduct model data) file format (PRATT 2001) is used for all 3D geometric models within the process. Additionally to the geometric data, some functional information about the models is required by several applications in the process. This data is, for example, which geometric entities of the WP are used as locating, supporting and clamping entities in a setup. A way to provide such data in geometric models is the use of annotations. New application protocols of the STEP format, like AP 203 ED2, allow for the addition and storing of annotations with the geometric model (INTERNATIONAL STANDARDIZATION ORGANIZATION 2011). However, *OCCT*, the open-source framework used in this approach for the implementation of all applications that require geometry handling, does not support any of the application protocols of the STEP format that allow for the addition of annotations. Until this is supported by *OCCT*, a workaround is necessary.

Using the shape grammar interpreter (SGI) module *Spapper*, 3D labels can be stored together with 3D geometry in *FreeCAD's* native file format FCStd. These labels can be used like annotations to store semantic data in the form of text and spatial positions. FCStd files are a zipped collection of geometric data files, in the BREP format, and XML files that store the meta-data of the geometric entities (FREECADWEB.ORG 2014). All data of *Spapper* 3D labels is stored in these XML files. As XML is a common file format that can be easily processed computationally, the FCStd format (and use of *Spapper* 3D labels) is chosen for the representation of both geometric models and semantic annotations. Unlike the STEP format the FCStd format is not an industry standard. Therefore, the STEP format is used in the approach as the primary data format whenever possible and the FCStd format is used additionally when semantic data is necessary.

The approaches, software tools and prototypes discussed above are evaluated by processing a set of case-study parts and analyzing the results. In the next section, the list of case-study parts and the manual setup planning procedure are described.

## 3.8 Setup Planning and Parts Used for Verification

To validate the overall approach and all developed applications a set of demo parts is used. These parts stem either from the CogMaSh project or from the Drexel design repository (DREXEL UNIVERSITY 2011). The repository was scanned for 3D models of parts that can be produced using 2.5D milling and that meet the geometric restrictions imposed in Section 1.5. Some of the chosen designs were scaled proportionally (up or down) to fit the clamping length

range of the flexible vise. Table 3-2 gives an excerpt of the case study parts. The complete list can be found in Appendix 10.2.

In each line of Table 3-2 the part name, the unique part ID, the maximum dimensions of the blank and the total number of setups required is listed. The model of the part, of the blank used to machine the part from and, if required, of the intermediate WP shape is shown. The intermediate state serves as WP for a subsequent setup. Thus, intermediate states only exist for parts that require more than one setup to yield the final shape from the initial blank. Further, the contact surfaces of the WP for the respective setup are shown in the "Setup" columns. These are the surfaces that are used for supporting, locating and clamping the WP. As this is an approach for low volume production, forged or casted blanks in a close-to-final shape are unlikely to be used. Box-shaped and cylindrical stock material is used, because these are common blank shapes in this manufacturing scenario. However, the intermediate WP states can have more complicated outer contours that need to be located and clamped. As mentioned in the introduction, the scope of WP shapes for the approach is limited. The geometry of the WP must fulfill the following criteria: All contact surfaces have to be either plane or cylindrical. Only cylindrical surfaces, i.e. curved surfaces with constant radii, are considered, while freeform surfaces or curved surfaces with non-constant radii are not. With the given vertical three-axis machine, the active surface, the surface that is to be machined, has to face upwards when the WP is positioned in the fixture. One or two WP surfaces can be used to vertically locate the WP. Unlike the top surface, these bottom surfaces have to be horizontal, i.e. parallel to the machine tool table. As only 2.5D milling can be used for the machining of the jaws and as the machine is not equipped with angled face milling tools, no angled contact jaw surfaces (non-vertical or non-horizontal) can be produced. Thus, the side surfaces of the WP have to be perpendicular to the bottom surface(s). When oriented, the WP can have a maximum x-dimension of 140 mm, maximum y-dimension of 140 mm and maximum z-dimension of 150 mm. If two different vertical locating surfaces are used they can only be 20 mm apart from each other in z-direction. For cylindrical WP surfaces, the maximum WP radius is 100 mm. These values are based on the vise geometry and machining envelope and are explained in more detail in Appendix 10.6.2. All parts and setups presented in Appendix 10.2 fulfill these geometric restrictions.

The selection of the blanks, the setup planning and corresponding NC code generation is carried out manually, as explained before. The setup planning is done by orienting a given 3D WP model (in *FreeCAD)*, such that the following conditions hold:

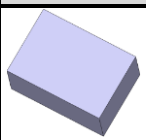
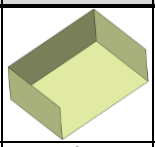
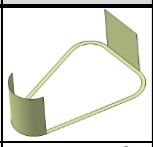
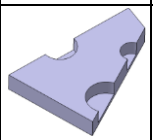
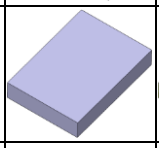
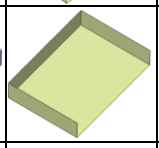
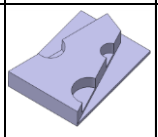
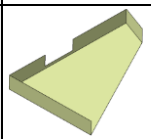
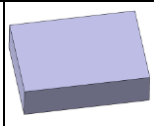
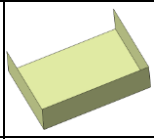
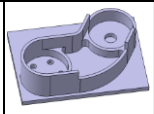
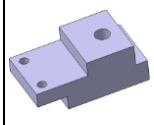
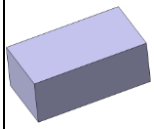
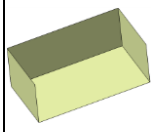
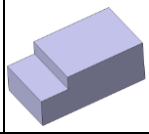
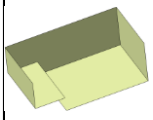
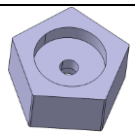
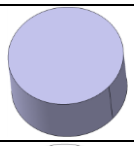
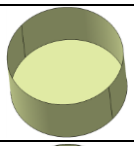
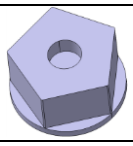
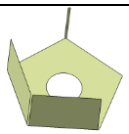
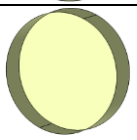
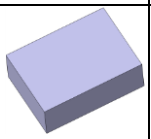
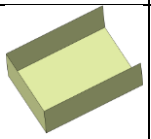
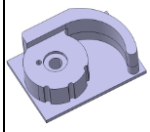
- The active surface of the WP has to face upwards.
- The WP must have a planar, horizontal surface facing downwards as primary locating surface.
- The WP can have another planar, horizontal surface facing downwards as supporting surface.
- The WP must have vertical secondary/tertiary locating and clamping surfaces of planar or cylindrical shape.

- The distance of the secondary locating surface and the clamping surface cannot be bigger than the maximum clamping width of 140mm.
- The secondary locating surface must have a common edge with the primary locating surface.
- The clamping surface must have a common edge with the supporting surface.
- If the secondary locating surface is planar, a tertiary planar locating surface is required, which can be in a sharp, right or obtuse angle to the secondary locating surface.
- If multiple planar surfaces can be used as secondary locating surface, those that have a parallel clamping surface are to be preferred.
- If the secondary locating surface is cylindrical no tertiary locating surface is required.
- If the secondary locating surface and the clamping surface are cylindrical, the WP has to be rotated such that the center-points of the cylindrical surfaces are on a line parallel to the x-axis.
- If a cylindrical or planar surface can be used as the secondary locating surface, the cylindrical surface has to be chosen.
- Only a single surface is to be used for clamping, to not cause issues of over-determination.
- If the secondary locating surface is planar, the clamping surface should be parallel to it or a line contact should be used for clamping.

These rules represent common guidelines for setup planning and are respected in the setup schemes shown in Table 3-2. As outcome of the manual setup planning process each WP model is oriented in space such that 1) the primary locating and supporting surfaces are parallel to the x-y plane, 2) an outward normal of the secondary locating surface is facing in the negative x-direction of the vise and 3) an outward normal of the clamping surface is facing in the positive x-direction and is on the same axis as the normal of the secondary locating surface. The placement of the WP, i.e. the translational parameters, can be arbitrary.

The presented set of case-study parts covers all WP shapes that are within the scope of the approach and is, hence, adequate to evaluate the given approach. An increase in part complexity, while respecting the limitations of the part geometry imposed through the scope of this work, would only lead to a higher number of necessary setups for one part but not to more challenging examples.

**Table 3-2 Excerpt of the list of case study parts used for verification including models of blanks and intermediate workpiece models**

| Name | Part ID | Dim. blank | # of setups | Part | Blank | Setup 1 | Inter-mediate_1 | Setup 2 | Source |
|---|---|---|---|---|---|---|---|---|---|
| Allied Signal ACIS09 | 9001 | 110x 75x 45mm | 2 |  |  |  |  |  | Drexel Design Repository |
| Angled Socket | 9015 | 98x 70x 18mm | 2 |  |  |  |  |  | CogMaSh |
| Cap Arm | 9017 | 126x 90x 35mm | 2 |  |  |  |  |  | Drexel Design Repository |
| Cube_CS 75x 40x30 | 9005 | 75x 40x 30mm | 2 |  |  |  |  |  | CogMaSh |
| Pentagon | 9016 | D68x 35mm | 2 |  |  |  |  |  | CogMaSh |
| Cylinder D60_ H15 | 9008 | D60x 16mm | 1 |  |  |  | | | CogMaSh |
| Lever | 9019 | 123x 88x 45mm | 2 |  |  |  |  |  | Drexel Design Repository |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

## 3.9 Expected Research Contribution

Based on reaching the research goals within the presented approach and taking the shortcomings of existing related work into account, as presented in Section 2.4, the following research contributions are expected from the work presented in this thesis:

- Development of an ontology as a formal FD representation and integrated information model. Unlike existing fixture design ontologies the one developed has to be capable of representing vise-type fixtures and related fixture design knowledge.
- Evaluation of the use of the developed formal FD representation for (semi-)automatic fixture configuration design based on semantic reasoning, i.e. the determination of suitable fixture components for a given fixture problem.

- Development of a spatial grammar for the synthesis of new fixture component designs.
- Evaluation of the capabilities to embed requirements of a real, detailed engineering design problem in a spatial grammar to generate production-ready fixture component designs.
- Evaluation of the spatial grammar interpreter *Spapper* for the development and automatic application of grammars for detailed design tasks.
- Development of an automatic collision detection method and tool based on the tool-swept volume interference approach that can derive paths for each cutting tool used directly from the NC file of a part and is reusable and extendable for other scenarios.

A discussion of the achieved research contributions of this thesis is presented in Section 7.1.

# 4. Ontology for Fixture Design

In this section the development and resulting structure of an automated fixture design (AFD) ontology are presented. The ontology serves as a common fixture design (FD) representation, basis for communication among the FD applications and link between the AFD system and the manufacturing system. Further, the options for fixture configuration design through reasoning about the ontological knowledge representation, i.e. the generation of FD candidates using existing fixture components based on a given workpiece (WP), are evaluated.

In ontology modeling all objects, categories and relations are described as entities. In the sematic web language used to model the ontologies at hand categories are denoted as *classes*, objects as *individuals* and relations as *properties*. Properties can be further divided into *object properties* and *data properties*. Object properties relate one object to another while data properties relate an object to a data value (HITZLER et al. 2012, p. 4). Throughout the course of this work the terms are used interchangeably, e.g. category and class or relation and property.

## 4.1 Ontology Development

For the development of ontologies, several methodologies and processes have been described in literature. In this work the methodology depicted in Figure 4-1 is used. The methodology combines the knowledge meta-process presented by SURE & STUDER (2002) with the methods described by USCHOLD & GRUNINGER (1996) and STUCKENSCHMIDT (2009). By combining these approaches a complete development process with detailed methods for each step of the process is created. The knowledge meta-process consists of five main steps that are presented in Figure 4-1: feasibility study, kick-off, refinement, evaluation and application & evolution. The arrows show that building an ontology is a highly iterative process, with the insights created during later phases often affecting one or more of the preceding steps.

In the feasibility study the ability to model the domain knowledge in an ontology and the risks and chances of the model generation are explored. As it has already been shown in scientific literature that ontologies can be used to represent fixture design knowledge (AMERI & SUMMERS 2008; FAN 2010), feasibility is not considered in this work. In the kick-off phase, the requirements for the ontology are defined and documented and relevant sources of domain knowledge and existing ontologies of the domain are collected. Based on these sources, first basic entities of the ontology are identified and structured in an informal way, e.g. as a mind map. STUCKENSCHMIDT (2009) proposes the written definition of the purpose & scope of the ontology to document the requirements and the use of competency questions to focus the ontology and determine necessary entities.

In the refinement phase the initial structure and terminology created in the kick-off is extended and detailed. Using the competency questions and previously identified entities the process is carried out in a middle-out approach, where more entities are created based on concretization and generalization of the existing ones. Next, all possible class hierarchies or taxonomies are created and relations and properties identified to define the classes. The results should be evaluated constantly on an informal level, by checking the ontology against the competency

questions and the requirements, and on a formal level, as soon as the ontology is formalized. The formalization can be carried out right away or at the end of the refinement phase. The phase ends with an initial formal ontology of the domain.

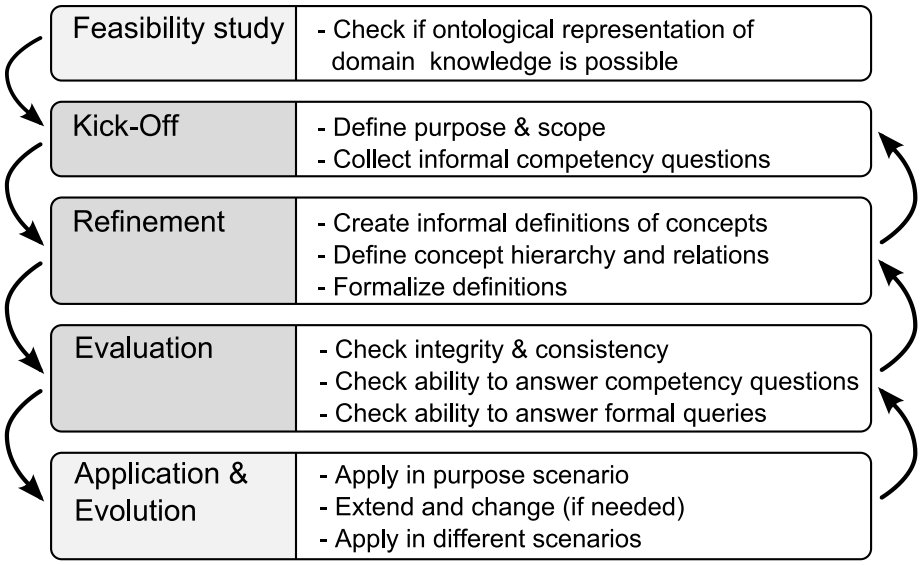



**Figure 4-1 Ontology development methodology with five iterative phases based on (SURE & STUDER 2002; USCHOLD & GRUNINGER 1996; STUCKENSCHMIDT 2009) as presented in (GMEINER & SHEA 2013a)**

Next is the evaluation phase, where the integrity and consistency of the initial ontology is evaluated by running a reasoner on the ontology to determine inconsistencies and inferences and by checking the ability to answer informal and formal queries. Queries for individuals require the presence of these in the ontology. Hence, before such formal queries can be issued the instantiation of a set of individuals, e.g. for WP and fixture components, is necessary. The ability to represent the individuals is also an integrity check. As mentioned, this phase is the most iterative one. Wrong results or the inability to answer queries and questions force a refinement of the ontology.

Once the evaluation phase is successfully finished, the now fully formalized ontology can be applied as defined in the kick-off phase. Using a reasoner and formal queries the ability of the ontology to represent all required knowledge is tested, and if needed, the ontology is again refined and evaluated to extend it. The evolution of the ontology requires long-term use in practice, preferably in different scenarios and by different domain experts, and includes issues like defining a maintenance and update plan. As this is outside the scope of this work, the evolution is not included.

### 4.1.1 Ontology Kick-Off

During the kick-off phase the purpose and scope of the ontology are defined. In USCHOLD & GRUNINGER (1996), the use of a motivating scenario is suggested. Although one motivating scenario cannot cover all potential use-cases of the ontology, it is a solid starting point to define

the requirements of the ontology. The scenario that sparked the development of the ontology is as follows:

The manufacturing system is given a job to produce a new, non-predefined part. To match the WP shape and dimensions, the fixture needs to be automatically configured. For the presented reconfigurable fixture device to be economical, existing jaws are to be reused when possible. The purpose of the ontology is, thus, to serve as a fixture design knowledge-base, allow for communication of data between the different FD applications and the control system of the vise and enable fixture configuration design through knowledge inference based on reasoning.

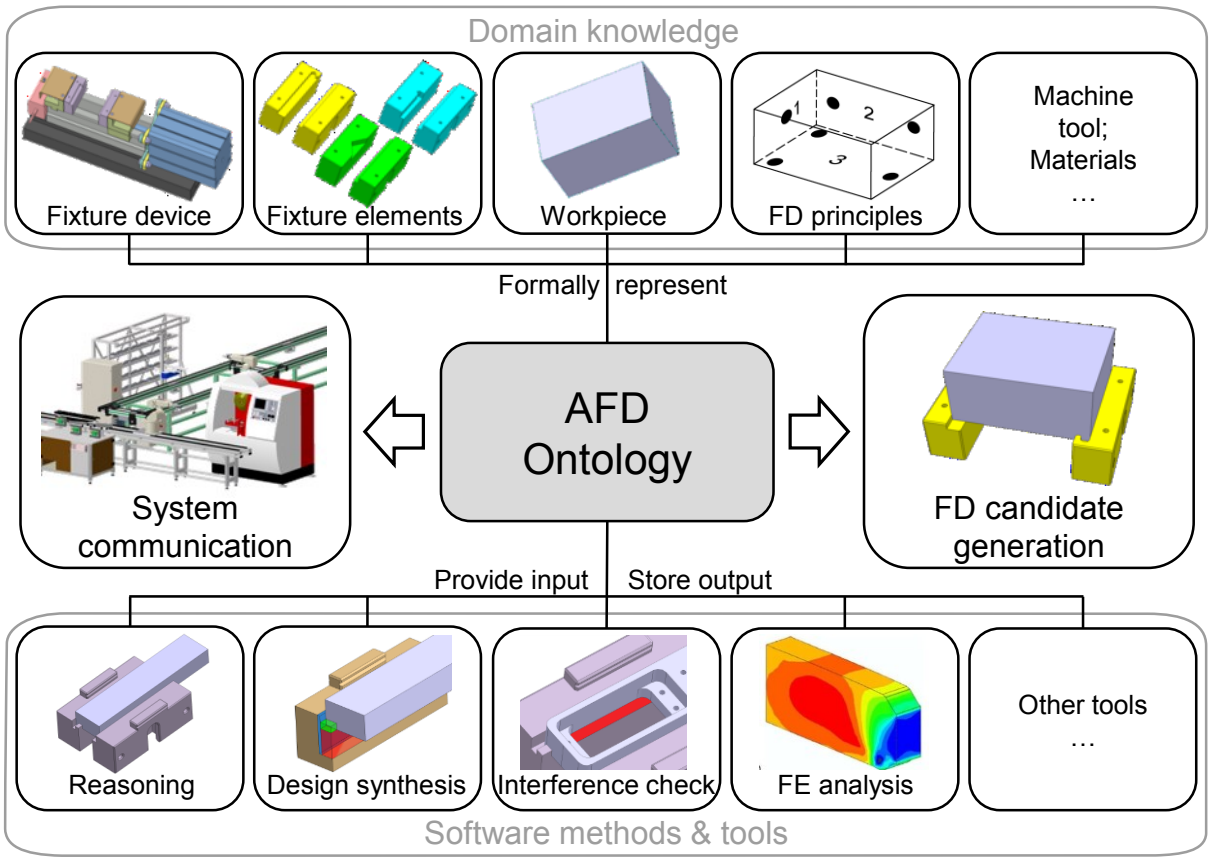



**Figure 4-2 Purpose and scope of the AFD ontology - Formally represent domain knowledge, serve as database for fixture design tools, allow for system communication and generation of fixture design candidates by reasoning, in analogy to (GMEINER & SHEA 2013a)**

The scope of the ontology is to represent FDs including the fixture components and the WPs on a semantic level. Knowledge or data specific to the fixture device and the machine tool, e.g. the maximum clamping width or the machining envelope, and job specific data like machining forces and material data also needs to be formally represented and defined within the ontology. The general scope presented in Section 1.5 also has to be considered. Setup planning knowledge or the representation of freeform surfaces is, therefore, not covered in the ontology. Figure 4-2 graphically explains the purpose and scope of the ontology.

After defining the purpose and scope a set of competency questions is developed. Competency questions are informal queries that the ontology must be able to answer. These questions help to focus the ontology and determine the basic classes and relations. Additionally, the competency questions can be used to evaluate the integrity, correctness and consistency of a created ontology by checking if all questions can be answered (STUCKENSCHMIDT 2009; USCHOLD & GRUNINGER 1996). To cover the domain as comprehensively as possible an initial set of competency questions is written out and then extended. For all initial questions, the rationale (one level higher: How is the solution used?) and the decomposition (one level lower: What solutions are necessary to answer the question?) is determined. Figure 4-3 shows the procedure for one competency question.

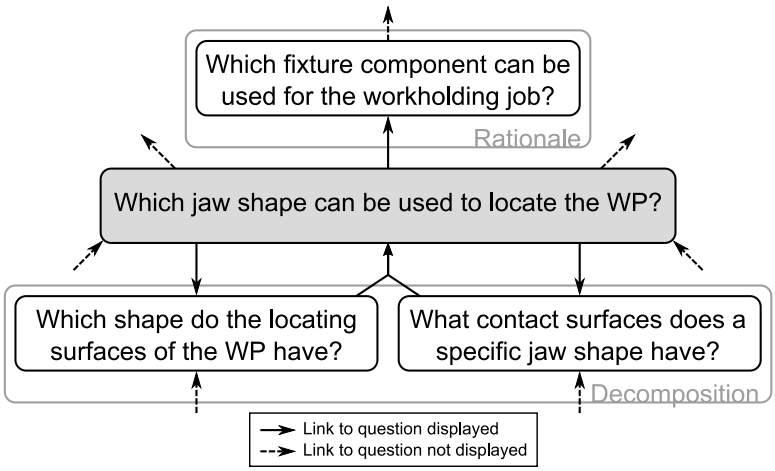



**Figure 4-3 Example of an informal competency question with rationale and decomposition for the development of the AFD ontology**

To answer the question which jaw shape (or design) can be used to locate a given WP (Figure 4-3 middle) one needs to know, amongst other things, which shape the locating surfaces of the WP have and what types of contact a specific jaw shape has (Figure 4-3 bottom). These questions can be generated by decomposition. The answer to the middle question is part of the information required to determine which fixture component can be used for a given workholding job, i.e. for the WP (Figure 4-3 top). Thus, this question is a rationale. The example shows that one question can require the answers to several lower level questions. Further, the answer to a competency question can also serve as an input to more than one higher level question. The outcome of this procedure is a more detailed, linked list of competency questions. However, this list is not likely to be complete, i.e. representing the whole domain, as the evaluation and application of the ontology often expose missing questions. This can cause an iterative loop in the process, as described before.

Once the set of competency question has been created, all relevant entities, i.e. objects, categories and relations, for the ontology are identified, based on the words used to form the questions. This list is then extended by terms from fixture design and CAFD literature (HOFFMAN 2004; BI & ZHANG 2001; PEROVIĆ 2013; CHANG et al. 1998) and terms used in the

existing ontologies of the domain (AMERI & SUMMERS 2008; FAN 2010). The result is a multitude of entities that should be represented or at least consider in the development of the ontology.

### 4.1.2 Ontology Refinement

In the refinement phase, the identified entities are structured. The first step is the creation of class hierarchies. A class hierarchy must be based on domain knowledge and consists of classes and subclasses connected via *isA* relations. The goal of this step is to determine all classes and hierarchies among them from the list of entities. Figure 4-4 shows the hierarchy of the class *Fixture,* which is one of the classes resulting from the process. Fixtures are generally divided by their type (*FlexibleFixture* or *DedicatedFixture*) or by the manufacturing process they are used for (*MachiningFixture, AssemblyFixture,...*). Thus, these are the subclasses of *Fixture*, which again can have further subclasses, e.g. of different types of flexible fixtures.

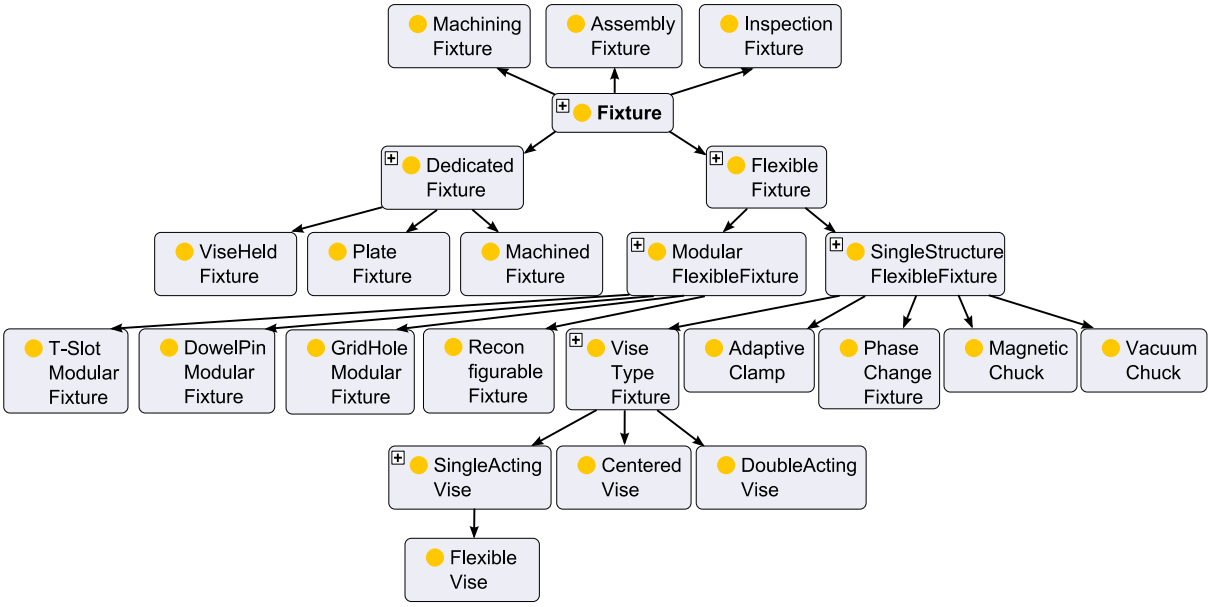



**Figure 4-4 Hierarchy of the class Fixture with 'isA' relationships between subclasses and superclasses**

After the concept hierarchies are created, the remaining entities are being checked for relations that link the classes and for data properties that describe them. Relations, or object properties, and data properties are a prerequisite for the formalization of the ontology through sufficient and necessary constraints. To identify relevant relations, the characteristics of the entities are checked. If categories or objects can be differentiated from each other through a characteristic, then it can be used as a relation. When relevant relations are found, their link to the defined classes, i.e. the domain and range of classes they connect, can be determined (STUCKENSCHMIDT 2009) and later explicitly defined in the formalization. The relations created this way are semantically rich connections between the classes presented in the concept hierarchy and can be used to form axioms that define the classes.

As an example, the hierarchy shown in Figure 4-4 can be used. Each *FlexibleFixture* can be differentiated from a *DedicatedFixture* by the fixture components it is built from. Same goes for the different types of flexible or dedicated fixtures. For a *MachiningFixture*, *AssemblyFixture* or *InspectionFixture* this differentiation, however, cannot be used. It is, for example, possible that the same fixture is used as *MachiningFixtur*e and as *AssemblyFixture*. Here, the relation to a type of manufacturing process can be used to define the differences of the classes. Considering the ranges of the two relations the need for a *FixtureComponent* class and a *ManufacturingProcess* class can be derived. Depending on the fixture components it is built from and the manufacturing process it is used in, a specific fixture (an individual) can then be inferred to belong to certain fixture subclasses.

All terms that are common in the domain need general definitions that are agreed on by the domain community. This is necessary to assure and ease the reusability and extendibility of the ontology. The definitions can be captured informally, using text, or formally, using an ontology language. For the classes presented in the ontology of FAN (2010) no definitions or formalizations are given, hence they could not directly be reused. For the FIXON ontology the formalization is available. Further, informal definitions for the FIXON ontology are given in AMERI & SUMMERS (2008). As some of these definitions are found to not be detailed enough to represent FD on a level that allows for reasoning, they need to be reworked or extended. The goal of the presented work is the development of a formal ontology. Thus, only some core classes are defined informally, while the majority of the definitions are formalized right away. Table 4-1 gives examples of some of the informal definitions created. These textual definitions can help to better capture the domain and serve as a starting point for the formalization. However, the formalization process of informal definitions can be highly iterative as the formalization language does not provide the same expressivity as natural language. If an existing informal definition cannot be converted to a formal one directly, a refinement of the informal definition or the addition of further concepts becomes necessary.

Analyzing the textual definitions of the classes given in Table 4-1illustrates that an ontology is affected by the user-defined scope. For the presented scope, a fixture is a *Thing*, the highest-level term of ontologies. When thinking about a manufacturing system, a fixture would rather be considered a manufacturing system (sub-)component, i.e. a subclass. Especially when reusing existing ontologies, this has to be kept in mind.

Using the hierarchies and informal definitions the domain knowledge is formalized. All classes, definitions and relations are transferred into classes, axioms, object properties and data properties. The classes can be created directly as named classes or as anonymous classes, based on placing constraints on the class extension, such as property restrictions, the enumeration of individuals that belong to a class or Boolean operations. More details on OWL 2 constructs and axioms are given in HITZLER et al. (2012).

**Table 4-1 Examples of informal class definitions of the AFD ontology**

| Class | Informal definition |
|---|---|
| Workpiece | A thing with a solid geometry and functional geometric entities, such as features or surfaces, that are used for locating, supporting, clamping and are defined by the setup plan. |
| Fixture | A thing built from fixture components and used in a manufacturing process to hold a workpiece in a defined setup by locating and maybe by supporting and clamping it. |
| Locating entity | A geometric entity, such as a surface or line, of a workpiece that is subject to contact with a fixture component that has a locating function. |
| Solid | A geometric entity of some fixture component or workpiece. Every solid has a 3D model path and can have a labeled model path. |

At this point no examples of the formalization of classes, relations and individuals are given, as the formal definitions do not add to the understanding. For those interested, the ontology files expressed in OWL functional syntax are provided in an online ontology repository (GMEINER 2014). The resulting ontologies and underlying modeling thoughts are presented in the following sections.

## 4.2 Ontological Representations for the AFD System

As a result from the development process and the idea of facilitating the reuse and extension of the created ontologies, two formal ontologies are created. The first ontology is a loosely defined upper ontology that classifies fixtures and fixture components and can be reused and extended in other fixture scenarios or other domains, e.g. for manufacturing system or process models. The second ontology, which imports and builds upon the upper fixture ontology, is a detailed fixture design ontology for the AFD system of the flexible vise. This ontology is modeled with a focus on enabling inference of knowledge about possible fixture configurations for a given fixture problem by reasoning and is scenario and device specific.

The AFD ontology should serve as integrated information model for the complete FD process it also imports an existing material ontology (CENTER FOR EDESIGN 2005) that covers the material data required in the FD verification phase. The FIXON ontology is not directly imported but the conceptual ideas are included in the modeling of the fixture design ontology.

### 4.2.1 Fixture Ontology

The fixture ontology is a basis of the fixture design ontology. It classifies different fixture types and fixture components. The ontology covers many common types of fixtures and is not limited to the flexible vise. The classification is based on literature (HOFFMAN 2004; BI & ZHANG 2001) and the classifications used in FIXON (AMERI & SUMMERS 2008) and the FD ontology of FAN (2010).

The fixture ontology is an upper ontology in the form of a class hierarchy which only uses necessary conditions for the description of the classes. Classes modeled in this way are called primitive classes, as opposed to defined classes, which are based on necessary and sufficient descriptions. While the use of primitive classes does not allow for advanced knowledge inference through reasoners, it facilitates the reuse of the ontology as part of other ontologies. Generally, the more rigorously defined the concepts of an ontology are, the more knowledge can be inferred. However, because of the detailed specifications it also becomes harder to consistently integrate it in other ontologies. Thus, the fixture ontology is kept at a high level and the necessary conditions are transferred into sufficient conditions or sufficient conditions are added after the import in the fixture design ontology.

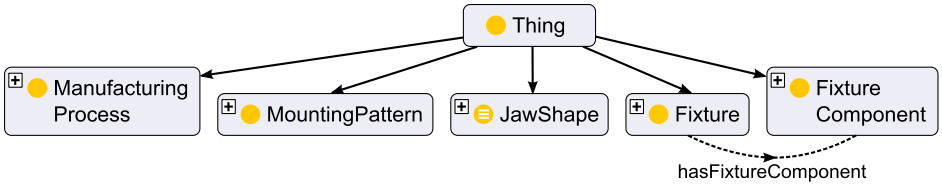



**Figure 4-5 First level of classes (neighborhood of class Thing) in the fixture ontology**

Figure 4-5 shows the main structure of the fixture ontology. The root classes are *Fixture*, *FixtureComponent*, *JawShape,* *MountingPattern* and *ManufacturingProcess*. The *FixtureComponent* class, shown in Figure 4-6, includes subclasses for common components, such as *FixtureBody*, *Locator*, *Clamp* or *Jaw*, which again have further subclasses, as shown for the *Jaw* class. Each class inherits the descriptions of its superclass, meaning that a subclass always has to meet the description of the superclass. Note that the *subclassOf* or *isA* relation, indicated by the solid lines in the figures, is a necessary condition. Additionally to the *subclassOf* relation, some object properties, like *hasJawShape*, *hasMountingPattern*, *hasFixtureComponent* or the inverse *isFixtureComponentOf* are used in the class descriptions.

The hierarchy for the *Fixture* class was already shown in Figure 4-4. In this class, the subclasses are mainly described by the *hasFixtureComponent* property. For example, a fixture that has a *FlexibleFixtureBody* is considered a *FlexibleFixture*, while a fixture that has a *DedicatedFixtureBody* is a *DedicatedFixture*. The number of involved components can also be used to differentiate fixture types. A *SingleActingVise* uses one *LocatingViseJaw* and one *ClampingViseJaw*, while a *DoubleActingVise* uses exactly one *LocatingViseJaw* and two *ClampingViseJaw* instances. As an example, the class structure and axioms for the class *FlexVise* and all its super-classes is shown in Figure 4-7.

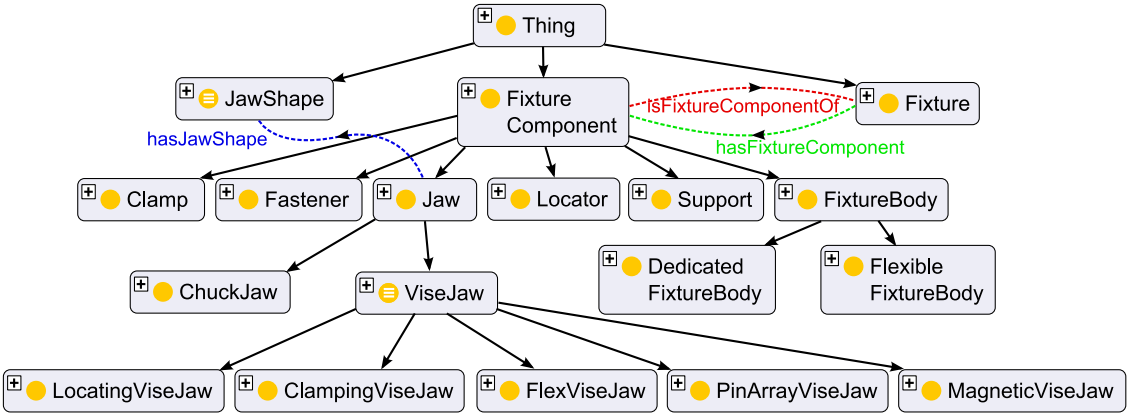



**Figure 4-6 Neighborhood of class FixtureComponent in the fixture ontology**

Following this modeling scheme, all common fixture types and components can be classified in the presented Fixture Ontology. If required, the ontology can also be extended with further classes. However, to go beyond the classification and allow for reasoning about fixture configurations, a more detailed fixture design ontology is required.

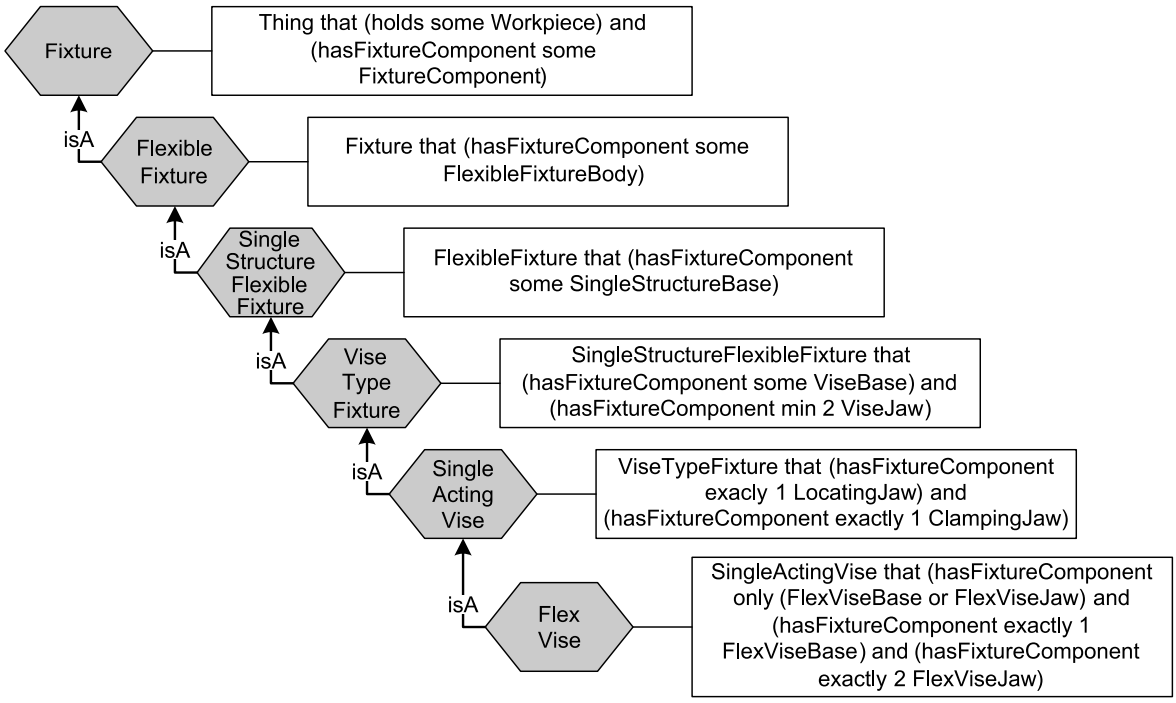



**Figure 4-7 The class FlexVise and its super-classes including the describing conditions (axioms)**

### 4.2.2 AFD Ontology for the Flexible Vise

Basis for the AFD ontology is the fixture ontology that is directly imported and then extended with sufficient conditions as well as classes, object and data properties to build an integrated information model that allows for knowledge inference. Figure 4-8 shows the main classes of the AFD ontology as subclasses of the superclass *Thing*. As can be seen, the bottom row of classes stems from the imported ontology while the top row of classes is added in the AFD ontology.

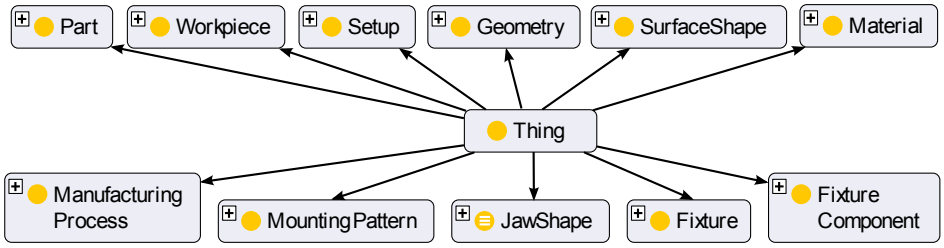



**Figure 4-8 First level of classes (neighborhood of class Thing) in the AFD ontology**

In Figure 4-9 the main classes are depicted schematically with the object properties that connect them. The property names were shortened to increase the visibility of the figure and the abbreviation FO illustrates that a class stems from the imported fixture ontology. A *Fixture*, built from *FixtureComponents*, is used to hold a *Workpiece*. To hold a WP implies that the fixture components *locates*, *supports* and *clamps* the specific WP. Each WP is linked to a *Setup* which in turn is linked to a *Part* and a *ManufacturingProcess*. A part can require multiple setups, but each setup only has one specific WP. Thus, a WP is just an initial, intermediate or final state of a part. Each part, WP and fixture component do have some *Solid* as *Geometry* and are made of a *Material*, which again has certain material properties. Further, the WP and the fixture components have some functional geometric entities represented via additional properties. For the WP these are the entities used for locating, clamping and supporting, as defined for the setup. These properties can be summed up under the term *hasSetupEntity*. In Figure 4-9 this relation is connected to the *Surface* class, to also show that every surface entity has a certain *SurfaceShape*. However, the ontology structure is not limited to the use of surfaces as setup entities. Lines and points from the geometry class can also be used. Additionally, the fixture components have functional entities from the geometry class. The *hasFunctionalEntity* relation links a fixture component to the geometric entities used for WP contact or assembly of the components. These entities can again be surfaces, lines or points.

Further, several data properties are used to add information to the classes or instances, which are not displayed in Figure 4-9. A set of numeric data properties is used for the description of all geometric entities in analogy to the Boundary Representation scheme (B-rep), which is a common formal geometry representation scheme used in CAD. These properties include the coordinates, length, width and vector directions of the geometric entities, e.g. to represent a surface via a point, length, width and outward normal. Another numeric data property adds a clamping force value to a specific fixture. Other data properties link strings to instances, e.g. to

represent the file path to the 3D model of a solid or to add a unique ID to a WP or fixture component.

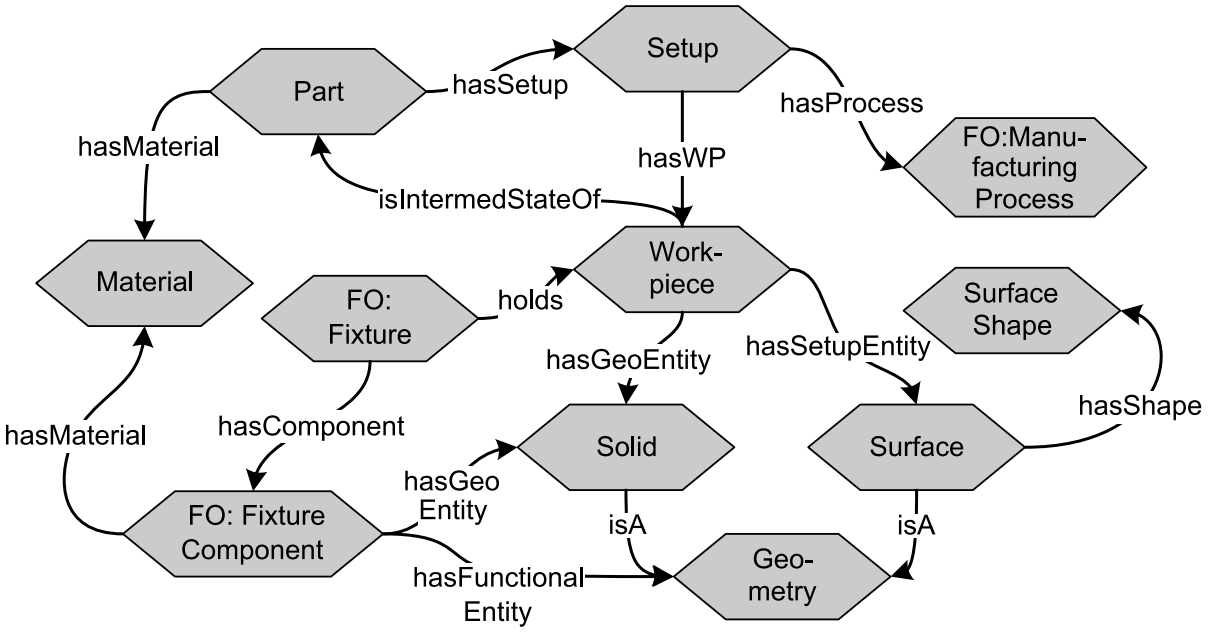



**Figure 4-9 Main structure of the AFD ontology – classes with object properties**

Many properties in the ontology are defined as functional and have defined domains or ranges. A functional property can only occur once for each object it relates, e.g. every WP can only have one specific clamping length. This does not restrict but allows for the detection of false definitions in the ontology by the reasoning engine. For example, if a WP is defined to have several clamping lengths, the ontology becomes inconsistent and a reasoner can deduce that this fact causes the inconsistency. Domains and ranges on properties are also not used as restrictions but as option for knowledge inference. For example, the *hasSetupEntity* property has the *Workpiece* class as domain and the *Geometry* class as range. Based on this definiton, a reasoner can infer that every concept that has a *hasSetupEntity* relation is a WP and every concept the relation binds to is a geometry object. Another method used in the modeling of the ontology is the propagation of information using property chains. For example, as the material of a part is the same for all WPs of that part, this information can be propagated. Every WP that is related to a part via the chain of a *isWorkpieceOf* and *isSetupOf* property inherits the *hasMaterial* property from the part. As a result, the material only needs to be defined for the part. All these modeling approaches increase the semantics represented in the ontology and reduce the need for explicitly modeled knowledge and, thus, the effort required for populating the ontology with data.

With the presented structure all core geometrical and functional aspects of fixture designs can be represented. The components used to build a fixture, the WP that the fixture holds, the geometric entities that belong to a spatial object and the function this entities have in the fixture design, the material details of the objects, the clamping forces, model paths and further details

can all be modeled in the AFD ontology. The positioning details of the components, and the mating relations between components and the WP can be used to describe a FD geometrically. For the fixture design tasks, only the "functional" geometric entities, e.g. the setup entities of the WP or the contact entities of the fixture components, are of importance and need to be modeled to allow for reasoning about feasible FDs, as explained later. However, the model structure allows for the representation of all geometric entities, as they can, e.g., be derived from a CAD model.

## 4.3 Fixture Configuration Design Based on Ontological Reasoning

One core aspect of using an ontology as formal fixture design representation is the capability of knowledge inference using reasoning engines. The goal is to automate the fixture configuration design process, i.e. to automatically generate FD candidates for a given fixture problem using the existing fixture components modeled in the ontology, as described in the FD process scheme shown in Figure 3-2.

The fixture problem is described by the WP and the setup definition, i.e. how the WP is oriented, which entities, such as surfaces or edges, of the WP are to be used for locating, supporting and clamping and how these entities are shaped and positioned. Based on this information the ontology should determine all fixtures that can be built from components in the knowledge base and can hold the WP. The link between the setup details of the WP and the capabilities of the fixture components to locate, support and clamp specific WP entities, is the basis for the fixture configuration design reasoning. As said before, the setup entities of the WP can be defined using properties. These setup properties are *hasPrimLocEntity*, *hasSecLocEntity* and *hasTertLocEntity* to represent the locating entities, *hasSuppEntity* for any additional supporting entity, *hasClampEntity* for the clamping entities and *hasActiveEntity* for the active entity. All of these properties link a WP to certain geometric entities, such as surfaces. The B-Rep style properties are used to describe the geometric entities in sufficient detail.

The link between the WP setup entities and the fixture components that can be used to be in contact with these entities are modeled in the ontology. This is done using complex class expressions and necessary conditions for these classes. For example, a class of all fixture component instances that have a V jaw shape can be defined. Given the ontology structure, the class expression written in the Protègè syntax looks as follows:

*Equivalent To:* *FixtureComponent that (hasJawShape value VJawShape)*

The reasoner infers that all instances that meet the class expression are members of the class. The class description can be extended with other necessary conditions, meaning that each member of the class is also inferred to fulfill these conditions. Coming back to the example, the V jaw class can be extended with the information that all members can locate a workpiece that has a cylindrical surface as secondary locating entity. The notation for this expression is:

*Subclass Of:* *locate some (Workpiece that (hasSecLocEntity some (Surface that hasSurfaceShape value CylindricalSurfaceShape)))*

Based on this description, the reasoner will infer that all members of the class, found via their jaw shape, also have the ability to locate WPs that meet the given geometric restriction. An illustration of this example is given in Figure 4-10. From the modeled data and using the given class expressions, it can be inferred that the instance *FlexViseJaw_0* is a *V_ViseJaw* that can locate the WP instance *WP_0*. Using this approach, basic locating, supporting or clamping capability information can be added to the fixture components.

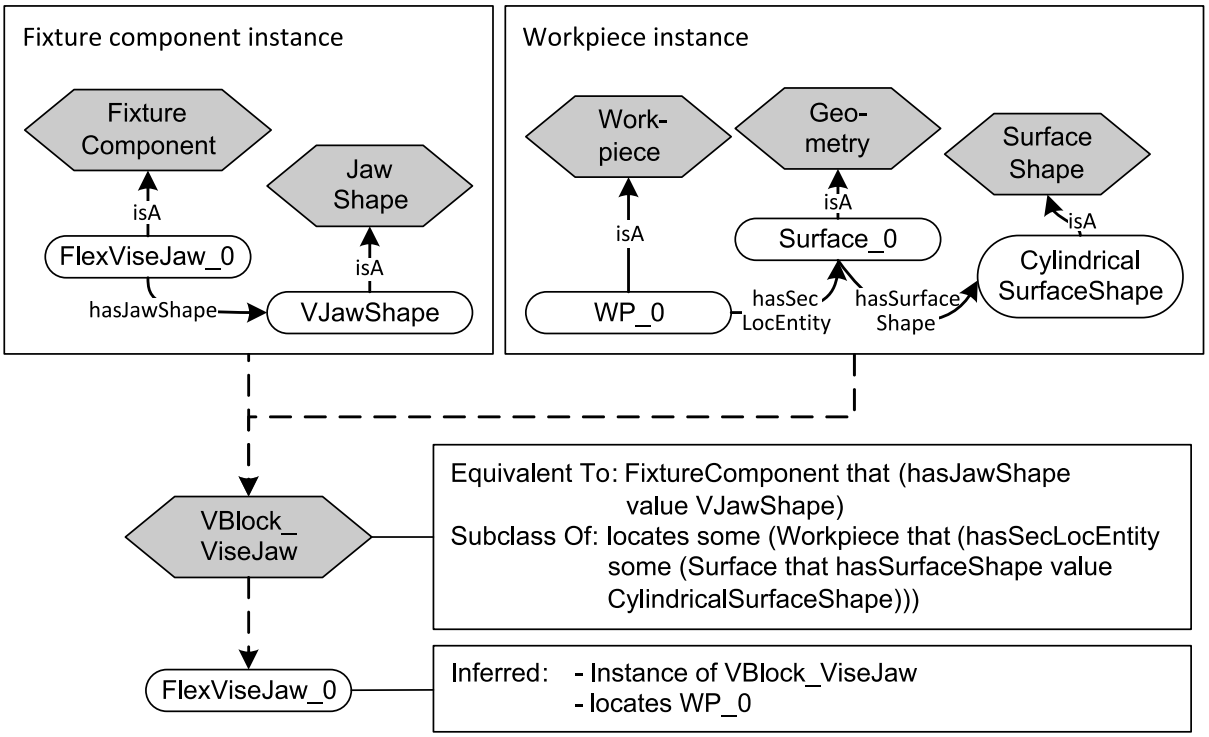



**Figure 4-10 Example of knowledge inference based on sufficient and necessary class expressions**

The given example is a simplification. In reality, the ability to locate, support or clamp a specific WP depends on more constraining factors that need to be modeled in the complex class expression. The final expressions are based on the 3-2-1 and the V-block locating principles, as described in Section 2.1.1. Using these expressions, the ability of each vise jaw shape to locate, support or clamp a WP of a certain shape is represented in the ontology.

Based on the added capabilities, the jaws can be classified into locating and clamping jaws. This allows for the generation of flexible vise fixture instances that follow the class definitions presented in Figure 4-7. According to this definition, each flexible vise fixture has to be built from one flexible vise base and two flexible vise jaws where one jaw has to be a locating jaw and the other a clamping jaw. Thus, a flexible vise instance needs to have *hasFixtureComponent* relations to one instance from each of the respective classes and only to these. To reason about the abilities of the flexible vise instances these need to be created. As the automatic generation of new instances is beyond the capabilities of Protégé or SWRL, the ontology pre-processing language (OPPL) is used for this (PALMISANO et al. 2013). OPPL is a scripting language for the processing of ontologies. It can be used to add explicit information to ontologies and also to

create new classes or instances. OPPL takes implicitly available knowledge inferred by the reasoner into account. The script used to create the flexible vise instances written in the OPPL syntax reads as follows:

*?fvb:INDIVIDUAL, ?lj:INDIVIDUAL, ?cj:INDIVIDUAL,*
*?fix:INDIVIDUAL=create(?lj.RENDERING+"_"+?cj.RENDERING) SELECT ?fvb*
*instanceOf FlexViseBase, ?lj instanceOf LocatingViseJaw, ?cj instanceOf ClampingViseJaw*
*BEGIN ADD ?fix hasFixtureComponent ?fvb, ADD ?fix hasFixtureComponent ?lj, ADD ?fix*
*hasFixtureComponent ?cj, ADD ?fix instanceOf hasFixtureComponent only {?lj,?cj,?fvb}*
*END;*

This script creates new instances by combining a flexible vise base with one locating and one clamping vise jaw for a certain WP and adding the *hasFixtureComponent* properties and a closure axiom to the instances. The script, therefore, creates all flexible vise fixture instances that can be built with the fixture component stock present in the ontology and that meet the class definitions. This ensures that only locating jaws are used for locating and clamping jaws for clamping. Note that the script does not imply that the created instances are of the type *Fixture*, or even of the type *FlexibleVise*. This is not required as, based on the ontology and the created instance relations and the closure axiom, the reasoner can infer that the instances are flexible vise fixtures. Figure 4-11 shows a screenshot of a fixture instance created by the OPPL script after a reasoner was run. The instance is named after the jaws used and has three *hasFixtureComponent* property assertions, two to the flexible vise jaw instances *FVJ_StRPos1* and *FVJ_St1* and one to the flexible vise body instance *FlexViseBase_1*. Further, a closure axiom on the *hasFixtureComponent* property, denoted by the word *only*, is added. Based on this explicit data, the reasoner infers that the fixture is a *FlexibleVise*, as can be seen by the expression in the dashed box.

Running the OPPL script creates and adds the flexible vise instances to the ontology. The information if the jaws used for a flexible vise candidate can locate, support and clamp a WP with setup entities of a certain shape is encoded via the class definitions. However, the feasibility of using a certain fixture instance to hold a specific WP is not yet modeled in the ontology. This requires a combination of the information from the fixture components plus the check of further geometric constraints. These constraints are 1) the adherence of the maximum opening length of the vise, 2) the matching combination of primary locating and supporting surface height and 3) the stable placement of the WP in the fixture. 1) is defined by the clamping length of the WP in combination with the maximum opening length of the vise and the exact geometry of the jaws. 2) is defined by the height of the primary locating and supporting entities of the WP and the respective contact entities of the jaws. For example, if the same WP surface is used for primary locating and supporting, then the respective contact surfaces of the locating and clamping jaw must be on the same level. 3) is defined by the position of the center of mass of the WP and if this point is within the jaw width when the WP is in contact with the jaws. If this is not the case, the gravity forces could cause the WP to slide out of the fixture when the vise is not engaged.

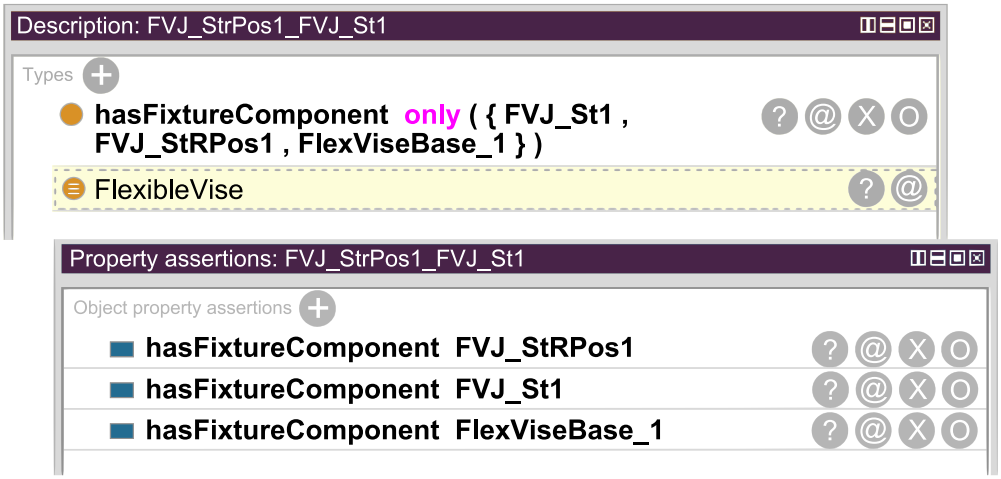



**Figure 4-11 A fixture instance with its property assertions created by the OPPL script**

Although the semantic web language OWL can be used to issue complex expressions, definitions that require data from several instances or mathematical operations with data values are beyond the expressivity of OWL. To extend the expressivity, the sematic web rule language (SWRL) can be used with OWL ontologies (HORROCKS et al. 2004). SWRL is supported by several OWL reasoners which can, therefore, be used to infer data based on the rules. SWRL rules allow, e.g., for the calculation of values based on data properties or the propagation of information from instance to instance. An example of a SWRL rule used to calculate the distance from the primary locating entity to the supporting entity of a WP and adding the resulting value as numeric data property to the WP instance reads as follows:

*hasPrimLocEntity(?wp, ?geo1), hasSuppEntity(?wp, ?geo2), hasPoint(?geo1, ?p1), hasPoint(?geo2, ?p2), hasZCoordinate(?p1, ?z1), hasZCoordinate(?p2, ?z2), subtract(?dist, ?z1, ?z2) -> hasPrimSuppDist(?wp, ?dist)*

In SWRL rules variables are noted with a preceding question mark. Every rule scans all instances in the ontology and for those that fulfill the left-hand side of the rule, implicitly add the right-hand side information, which is the part after the arrow. Implicit information equals data inferred by the reasoner. It is only present in the ontology as long as a reasoner is active. The shown rule determines all WP instances that have a *hasPrimLocEntity* and a *hasSuppEntity* relation to some geometry instance that has a point with a given z-coordinate, subtracts the coordinate values and uses the result as filler for the *hasPrimSuppDist* data property of the WP. Using this rule, the distance between the setup entities is implicitly added to the ontology when a reasoner is run, without the need to add the data explicitly. Thus, SWRL rules can also encode design knowledge and make the ontology semantically richer.

The implicit information generated this way can further be used in other SWRL rules, e.g. to constrain the fixtures that can hold a specific WP instance. The following statement is a

simplified version of the rule used to implicitly define a fixture that can hold an instantiated WP. The rule adds a WP instance as filler to the *holds* property of a fixture in case the fixture has a fixture component that locates the WP, a component that clamps the WP and the distance of the primary locating and supporting entities of the WP is equal to that of the respective jaw entities, denoted by the *hasStepDepth* property.

*Fixture(?fix), hasFixtureComponent(?fix, ?cj), hasFixtureComponent(?fix, ?lj), clamps(?cj, ?wp), locates(?lj, ?wp), hasPrimSuppDist(?wp, ?wpzdist), hasStepDepth(?cj, ?zcj), hasStepDepth(?lj, ?zlj), equal(?jawzdist, ?wpzdist), subtract(?jawzdist, ?zlj, ?zcj) -> holds(?fix, ?wp)*

Using a set of SWRL rules and a reasoner that supports those rules, the information about which fixture design candidates generated with the OPPL script can feasibly hold a specific WP is added to the AFD ontology. This is illustrated in Figure 4-12. A comparison to the description shown in Figure 4-11 shows that the *holds* WP property is implicitly added based on the SWRL rule. This way, the fixture configuration design for a workholding problem using existing fixture components can be automated. The determined FD candidates can also be extended with information like the clamping forces and be verified in subsequent FD process steps. An evaluation of the ontologies capability to serve as fixture configuration design tool and as integrated information model is presented next.

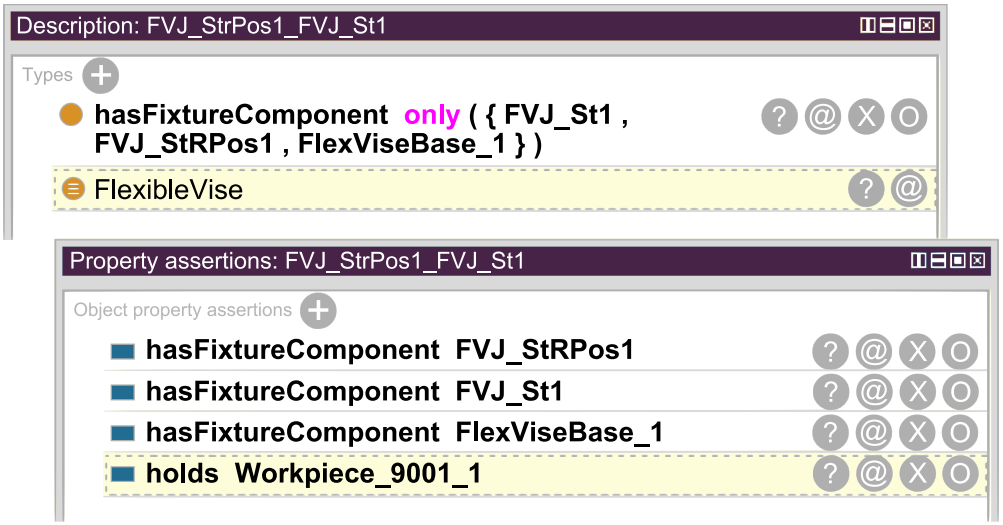



**Figure 4-12 A fixture instance with inferred holds Workpiece property created by the SWRL rules**

## 4.4 Evaluation of the AFD Ontology

To evaluate the AFD ontology as a knowledge base and fixture configuration design tool for the flexible vise, the ability to answer the competency questions, especially for WP-specific

fixture design candidates, are explored. For this, a set of fixture components needs to be instantiated. As according to the ontological definition each flexible vise fixture consists of a flexible vise base and two flexible vise jaws, instances of these classes are required to build fixture instances from. A single instance of the vise base is sufficient, as it is not subject to alteration. The ontology, however, allows for the existence of multiple flexible vise base instances to represent scenarios where several fixture devices are used in a manufacturing system or a company. Further, eleven different instances of flexible vise jaws are instantiated. Table 4-2 gives an excerpt of the jaw instances and the design information that is explicitly added to the ontology. The information in grey is not essential for the reasoning process presented. The complete table can be found in Appendix 10.3.1.

**Table 4-2 Excerpt of jaw instances and jaw design data explicitly added to the ontology for the evaluation (complete table in Appendix 10.3.1)**

| Flex Vise Jaw | hasID | hasJawShape | hasStep Depth | hasStep Width | hasStep Length | hasGeometricEntity/ hasContactEntity |
|---|---|---|---|---|---|---|
| FVJ_St01 | 3001 | Step | 2 | 100 | 4 | Solid_3001 |
| | | | | | | Surface_3001_0 |
| | | | | | | Surface_3001_1 |
| FVJ_StR_Pos2 | 1002 | StepRestPos | 3 | 79 | 9 | Solid_1002 |
| | | | | | | Surface_1002_0 |
| | | | | | | Surface_1002_1 |
| | | | | | | Surface_1002_2 |
| FVJ_StR_Neg1 | 1003 | StepRestNeg | 3 | 84 | 7 | Solid_1003 |
| | | | | | | Surface_1003_0 |
| | | | | | | Surface_1003_1 |
| | | | | | | Surface_1003_2 |
| FVJ_V02 | 2002 | V | 13 | -- | 13 | Solid_2002 |
| | | | | | | Surface_2002_0 |
| | | | | | | Surface_2002_1 |
| | | | | | | Surface_2002_2 |

The core criterion for the differentiation of the vise jaws is their shape. Four different basic jaw shapes are used for the flexible vise: step-shaped, V-shaped and positive and negative step-rest-shaped jaws. The different jaw shapes can be used to either locate or clamp and support a WP, according to either the 3-2-1 or the V-block locating principle, as defined in Section 2.1.1. For each jaw shape, different parametric variations can exist. This is explained in more detail in Section 5. Figure 4-13 shows one example of each jaw shape type that is used in the evaluation. In total, three different V-shaped jaws, four step-shaped jaws, two positive and two negative step-rest jaws are instantiated in the ontology for the evaluation.

V-shaped jaws can serve as locating or clamping jaws if the WP has cylindrical secondary locating or clamping surfaces. Note that cylindrical WP clamping surfaces allow for the use of either step-shaped or V-shaped clamping jaws, while planar WP clamping surfaces require step-shaped clamping jaws. Thus, V-shaped locating jaws can be combined with either step-shaped or V-shaped clamping jaws, depending on the shape of the clamping surface. These constraints

are embedded in the SWRL rule used to add properties to the jaw instances. When queried, the ontology should, therefore, only return fixture configuration designs with jaw combinations that match the WP shape. This illustrates that the fixture problem is described by the WP geometry and the setup data. This data either has to be explicitly added to the ontology or it must be implicitly given in the query. For the evaluation, possible fixture configurations for four different WPs are searched. The four WPs cover the geometric WP variations that the approach is focused on. They include planar and cylindrical locating and clamping surfaces and the use of different surfaces for primary locating and supporting. Figure 4-14 shows the first two WPs that represent the intermediate states of part AS_ACIS09, as presented in Appendix 10.2. The figure shows both the 3D models (top) as well as setup entities (bottom).

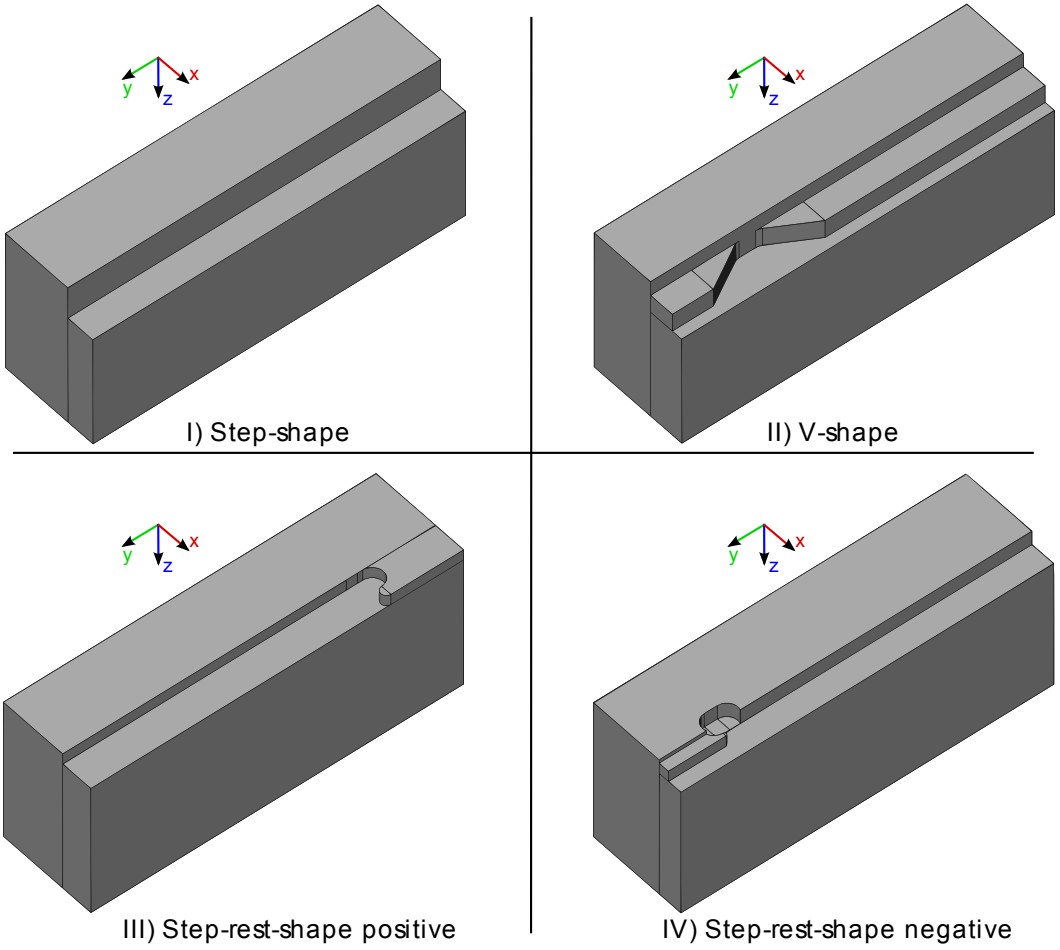



**Figure 4-13 3D models of vise jaws with different jaw shapes**

The first WP has ID 9001_0 and is a simple cuboidal WP with all planar setup surfaces. This is the stock for part AS_ACIS09. For such a WP shape a combination of a step-rest-shaped locating and a step-shaped clamping jaw is required. As the tertiary locating surface of WP 9001_0 faces in negative y-direction, a positive step-rest-shaped jaw must be used. The primary locating and the supporting surface are the same. Therefore, the step depth of both jaws needs to be the same to hold the WP horizontally.

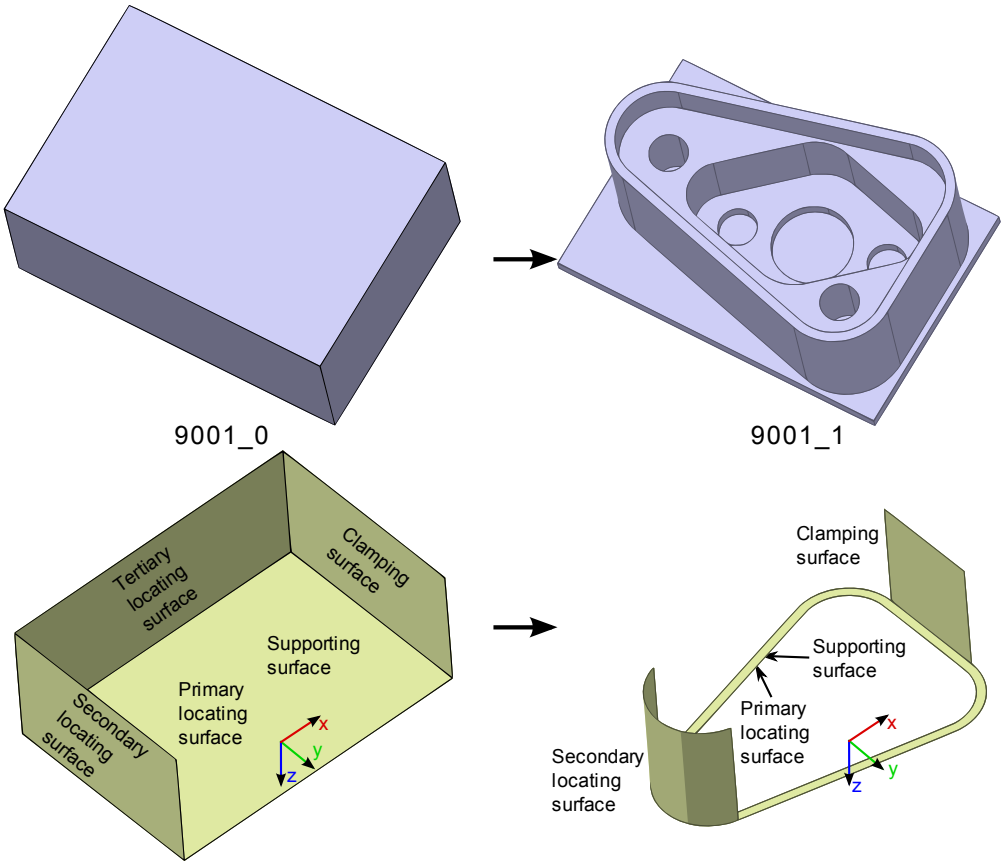



**Figure 4-14 3D models (top) and setup entities (bottom) of the workpieces 9001_0 and 9001_1, the intermediate states of demo part AS_ACIS09**

First, the reasoning capabilities without an instantiation of the WP data are tested. Thus, all required geometric information for retrieving fitting jaws must be given in the query. The queries are issued manually using the DL query tab provided by Protégé and it is selected that only instance results should be shown. Two initial queries are based on the competency question "Which fixture components can locate a WP?" and "Which fixture components can clamp a WP?". Screenshots of both query 1 and 2 and the results are shown in Figure 4-15. The first query returns all step-rest and all V-shaped vise jaws instantiated and the second query returns all step-shaped and V-shaped jaw instances. Because no other constraints are given in the query, all jaws that are inferred to belong to the locating and clamping jaw classes are returned.

To narrow down the selection of fixture components to the ones that meet the type and shape of the setup entities of WP 9001_0, the geometric information of the WP needs to be provided in the query. Thus, the next query represents the question "Which fixture components can locate a WP with a plane primary and plane secondary locating surface?". The query and the results are shown in the top part of Figure 4-16. This query returns only the step-rest-shaped jaw instances. However, it still returns the jaws with a negative step-rest-shape as result, which cannot be used for the given WP.

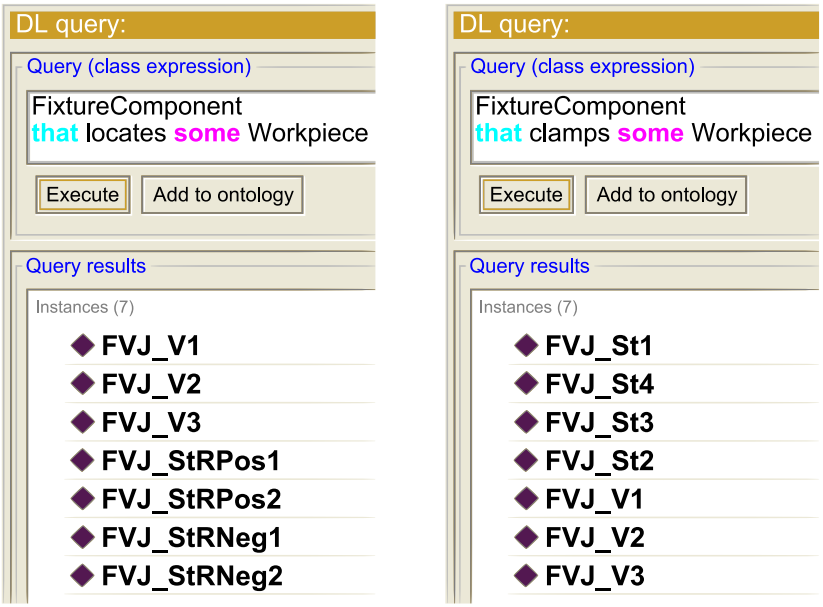



**Figure 4-15 Query examples 1 (left) and 2 (right) with resulting fixture component instances**

To allow for the determination of the correct type of step-rest-shaped jaws, the information about the y-direction of the normal vector of the tertiary WP locating entity needs to be added to the query, as shown in the bottom part of Figure 4-16. The competency question for this query is "Which fixture components can locate a WP with a plane primary and plane secondary locating surface and a tertiary locating surface facing in negative y-direction?". Now only the jaw instances which can theoretically be used as locating jaw for WP 9001_0, or any WP with a similar geometry of the setup entities, are shown. In the same manner, the correct clamping jaws, which are all step-shaped jaw instances, can be determined by giving the shape of the clamping and the supporting surfaces in the query.

So far, only queries for fixture components with certain abilities were issued. Note that the ability of the jaws to locate or clamp a WP was not added to the ontology via the instantiated jaw data, but that the jaw shape in combination with complex class descriptions allows for the inference of this information. As these queries only ask for fixture components and their capabilities, the results can be obtained without applying the OPPL script or the SWRL rules. However, besides the basic matching of the shape, the resulting jaw instances do not respect the other feasibility constraints discussed above, such as the stable WP placement or the difference of the supporting surface height. As these factors are defined by the interdependencies of the jaws, queries for feasible fixture candidates are required. The determination of results to such queries is also the main aspect of using ontological reasoning for fixture configuration design.

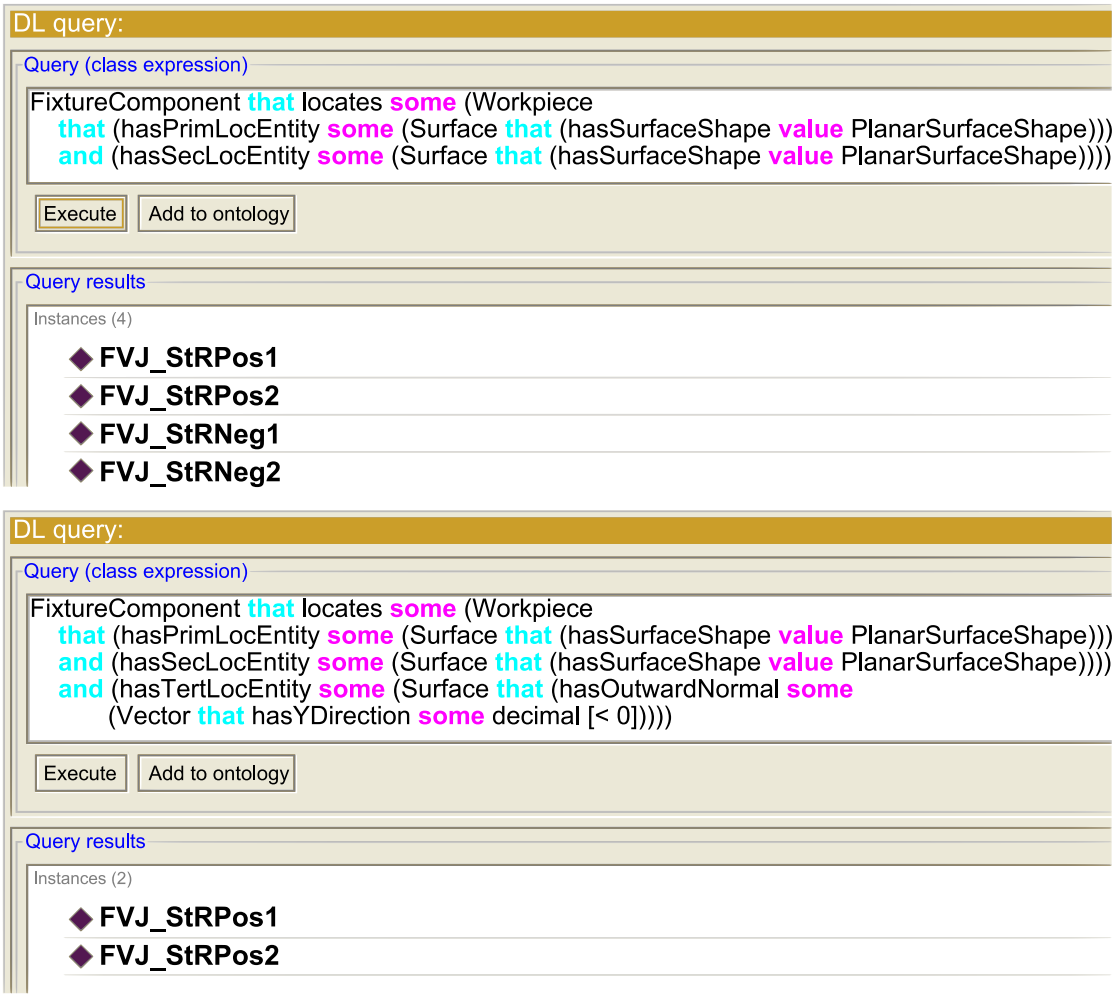



**Figure 4-16 Query examples 3 (top) and 4 (bottom) with resulting fixture component instances**

**Table 4-3 Excerpt of WP instances and according data explicitly added to the ontology for the evaluation (complete table in Appendix 10.3.2)**

| Work-piece | hasGeometricEntity/hasSetupEntity/ hasCoM | Setup function | (Surface) hasSurf-aceShape | (Surface) has OutwardNormal | (Vector) has Direction | (Surface) hasPoint | (Point) has Coordinate |
|---|---|---|---|---|---|---|---|
| 9001_0 | Solid_9001_0 | | | | | | |
| | Surface_9001_0_0 | PrimLoc, Supp | Planar | Vector_9001_0_0 | 0,0,1 | Point_9001_0_0 | (x,y,1044) |
| | Surface_9001_0_1 | SecLoc | Planar | Vector_9001_0_1 | (x,y,z) | Point_9001_0_1 | (1000,y,z) |
| | Surface_9001_0_2 | TertLoc | Planar | Vector_9001_0_2 | 0,-1,0 | Point_9001_0_2 | (x,962.5,z) |
| | Surface_9001_0_3 | Clamp | Planar | Vector_9001_0_3 | (x,y,z) | Point_9001_0_3 | (1110,y,z) |
| | Surface_9001_0_4 | Active | Planar | Vector_9001_0_4 | (x,y,z) | Point_9001_0_4 | (x,y,z) |
| | Point_9001_0_CoM | -- | -- | -- | -- | -- | (x,1000,z) |

To answer queries for feasible fixture candidates, the information added to the ontology by the OPPL script and the SWRL rules is required. By running the OPPL script with the instantiated fixture components 49 flexible vise instances are generated and added to the ontology. As the SWRL rules act upon the WP instances, queries for specific fixtures are not possible without instantiating a WP. Thus, the WPs used for evaluation are added to the AFD ontology at this point. An excerpt of the data of the WPs is given in Table 4-3. The complete table can be found in Appendix 10.3.2.

For each WP an instance is created. The instance is extended with property assertions such as *hasID*, *hasGeometricEntity, hasPrimLocEntity* or *hasClampingLength*. All instances that the properties should relate to, such as the *Surface* instances, need to be added, too. For the definition of the surface instances *Vector* and *Point* instances are required, and so on. This shows that adding the WP instance alone is not sufficient to model the required WP data. All entries given in black in Table 4-3 or Appendix 10.3.2 must be added to allow for proper reasoning. The information in grey is not essential for the reasoning process but can be added to the ontology to complete the information model.

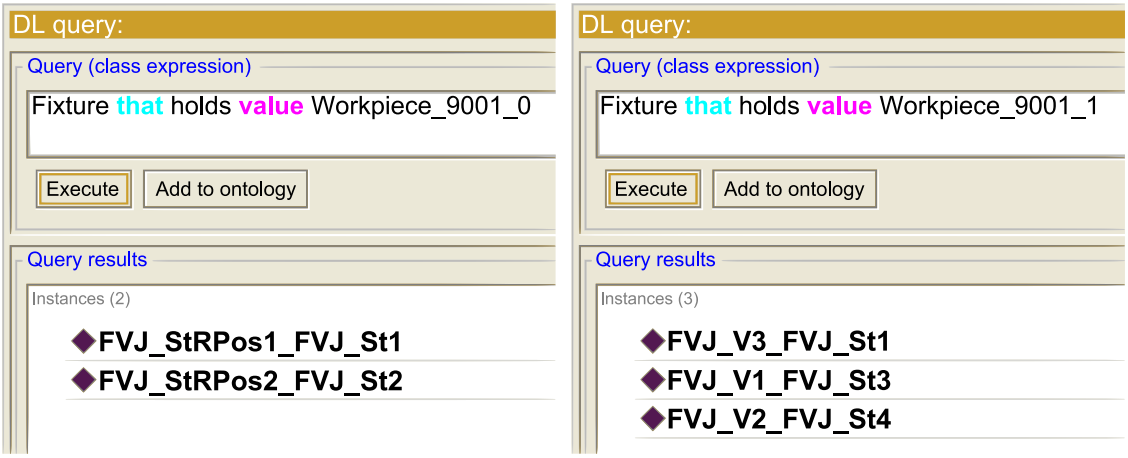



**Figure 4-17 Query examples 5 (left) and 6 (right) with resulting fixture instances**

Once this data is added to the ontology, queries about fixture design candidates that meet the feasibility criteria become possible. Figure 4-17 shows the queries and results for the questions "Which fixture candidates can hold WP 9001_0/WP 9001_1?". It can be seen that the queries return several but not all available instances of fixtures. For WP 9001_0 the query returns all combinations of jaws with a positive step-rest-shape for locating and jaws with a step-shape for clamping and supporting that also meet the other geometric design constraints. For example, as the WP uses the same plane surface for primary locating and supporting only jaw combinations with the same step depth are returned, but not the combinations with respectively offset surfaces, such as the fixture instance *FVJ_StRPos1_FVJ_St2*. In a similar manner all feasible combinations of V-shaped and step-shaped jaws are returned for WP 9001_1, which has a cylindrical secondary locating surface and a planar clamping surface.

The other two WPs used for the evaluation are shown in Figure 4-18. They represent the intermediate states of part CapArm, as presented in Appendix 10.2. The figure shows both the 3D models (top) as well as setup entities (bottom). The data for both WP, shown in Appendix 10.3.2, is instantiated in the ontology.

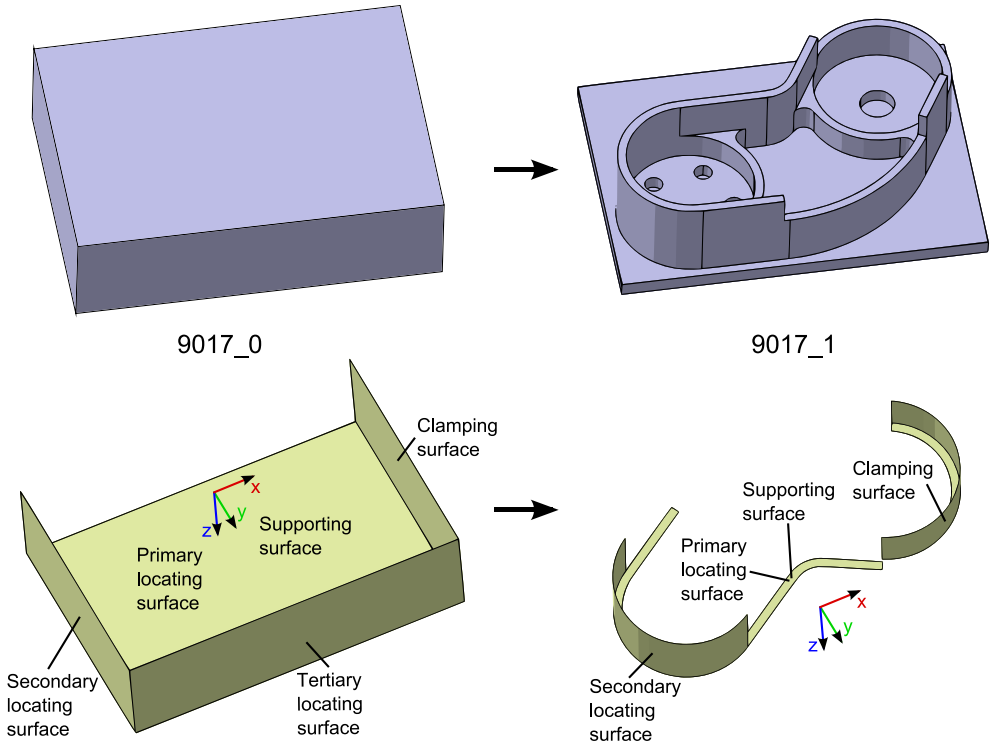



**Figure 4-18 3D models (top) and setup entities (bottom) of the workpieces 9017_0 and 9017_1, the intermediate states of demo part CapArm**

The first WP has ID 9017_0 and, like WP 9001_0, is a cuboidal WP with all planar setup surfaces. This WP, however, has a tertiary locating surface facing in the positive y-direction which requires a jaw with a negative step-rest-shape for locating. For clamping and supporting the WP, a step-shaped jaw is required. Further, the WP has a clamping length of 131mm, which is beyond the opening length limit of the vise. Thus, only jaw designs that provide the extra opening length through a high step length can be used for holding the WP.

The second WP with ID 9017_1 has a cylindrical secondary locating surface and also a cylindrical clamping surface. Thus, a V-shaped jaw is required for locating and both V-shaped or step-shaped jaws can be used for clamping. Further, this WP has different surfaces for primary locating and supporting, which are not in the same plane but differ in z-position by 10mm. Therefore, only jaw combinations that meet this 10mm offset are feasible.

Figure 4-19 shows the queries and results for fixtures that can hold these two WPs. A related competency question is “Which fixture candidates can hold WP 9017_0?”. Note that the queries are based on the ID and not on the name of the WP instance. Thus, the WP instance can have any name and only the unique ID of the WP has to be known to query the ontology for possible

fixture design candidates. This requires that the *hasID* data property is defined for the WP instances when they are instantiated. The results of the evaluation are discussed in the next section.

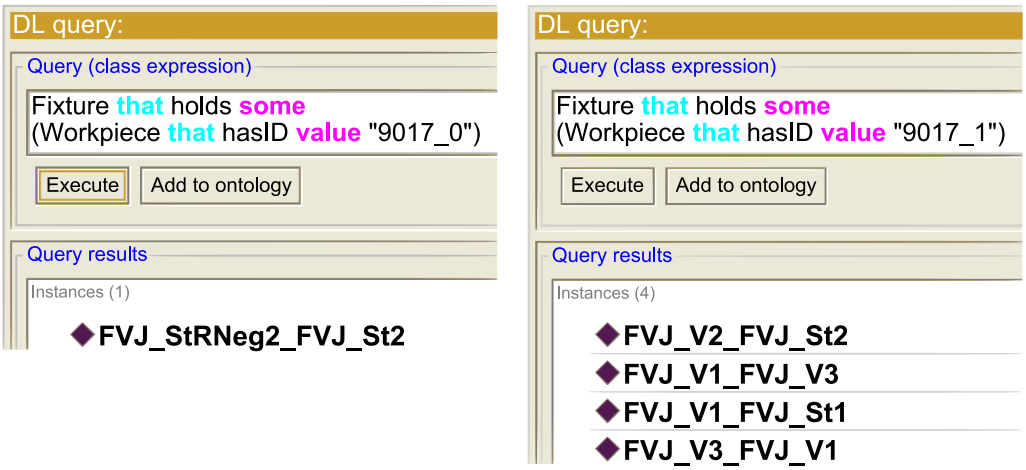



**Figure 4-19 Query examples 7 (left) and 8 (right) with resulting fixture instances**

## 4.5 Discussion

The evaluation shows that the AFD ontology is capable of representing fixture configuration designs on a semantic level and to answer the competency questions for WP-specific fixture components and fixture candidates. Other competency questions created in the kick-off phase can also be answered by the ontology but are not further discussed here. With the given ontological structure, a fixture is represented via the fixture components it is built from and the geometric details and fixture function of these components. The upper fixture ontology created and imported in the AFD ontology covers many different types of fixtures and fixture components and allows for their classification. The loosely defined hierarchy facilitates the reuse of this ontology (BORST 1997).

Similarly to fixture components, a WP is represented by its geometric entities, the use of these as setup entities and the geometrical data. SWRL rules and the geometric entities of a WP are used to enrich the present data by calculating additional data values, such as the clamping length or the z-distance of the primary locating and supporting entities. This decreases the need for data that has to be explicitly added to the ontology while still making the information available. A specific fixture design is then represented by a fixture used, the WP it holds and the clamping force applied. This allows for representing the reuse of the same fixture, i.e. one built from the same components, for different WP with different clamping forces.

Besides WP and fixture component data, the information required for the control of the fixture system, such as the opening length of the vise or the WP position in the fixture, as required by, e.g., a WP handling robot, are represented. To serve as data model for the complete FD process the representation allows to model the FD candidates generated by the computational synthesis

(see Section 5), includes the paths to all related models, as required by the assembly model generation and interference analysis (see Section 6.1) or the material parameters, as required by the FE analysis (see Section 6.2). The ontology can, therefore, serve as integrated information model for the complete FD process and the involved tools. As the focus of the ontology development was on a FD representation for the flexible vise that allows for reasoning about possible fixture configurations, not all input or output information for all involved applications or process steps are already included. FAN (2010) describes the integration of further FEM input and output data or the integration of features in the ontology, to add further information to the model. Other applications may require further geometric details, such as the bounding boxes required to define a Brep solid in case not all surfaces are modeled. Due to the extendibility of ontologies, adding such aspects to the presented AFD ontology can be carried out if needed.

Existing FD ontologies (AMERI & SUMMERS 2008; FAN 2010) represented FD mainly via the contact points between the fixture components and the WP in 3D space and, thus, only work for modular fixture systems with pin locators, supports and clamps. The representation used for the AFD ontology also covers this data but is not limited to points as geometric entities, as it can also cover lines and surfaces. Additionally, it includes knowledge about locating principles on a semantic level. This knowledge is included in the class descriptions of the fixture components and in a set of SWRL rules by modeling and matching the fixture functions of the geometric entities of both the fixture components and of the WPs. The ontology currently only covers the 3-2-1 and V-block locating principles (see chapter 2.1.1) for single-acting vise-type fixtures. Nevertheless, the generality of the representation scheme and the use of an upper fixture ontology that classifies many different types of fixtures and fixture components allows for an extension of the ontology to cover other locating principles and fixture types. This, however, has not been evaluated.

Besides the use of the ontology as an integrated information model, the evaluation further showed that, based on the AFD ontology, reasoning about feasible fixture configurations for a given workholding job is possible. All fixture candidates returned as answers to the queries meet the embedded design feasibility criteria without any exceptions. Complex class definitions and an OPPL script are used to generate all possible fixture configurations that can be built from the existing fixture components and SWRL rules are used to add the information which fixture configurations are feasible for specific WPs in a general manner. Several feasibility criteria are included in the reasoning process, such as a match of the WP setup entity shapes and the fixture component shapes or the adherence of geometric constraints like the maximum opening length of the vise. However, the generated fixture configurations still need to be verified in subsequent process steps as the feasibility check that is possible with the ontology is limited. Criteria like the occurrence of interferences between the fixture and the machining tools or checking the force equilibrium between the clamping, friction and machining forces are beyond the reasoning possibilities of an ontology. For these aspects, analysis methods are required.

The OPPL script creates explicit information, i.e. it permanently adds all fixture candidates that can be built with the existing components to the ontology. Thus, when new fixture components are added to the ontology all existing fixture instances need to be deleted and the OPPL script

needs to be run again. Further, once a final fixture candidate is selected after verification and evaluation and extended with the information created in these process steps, it should be renamed. This way, the extended information that transfers a fixture into a specific fixture design, such as the clamping force used, is permanently stored and not altered when the OPPL script is run again.

Looking at the automation of the knowledge modeling and reasoning process it was shown that the information about the workholding job and the available fixture components has to be present in the ontology to be able to reason about it. Although, as shown in Figure 4-15, a certain extent of reasoning is possible by giving the WP data in the query, not all queries are possible this way. To make full use of the reasoning capabilities, the fixture components (jaws), WP and the related data need to be added to the ontology by creating respective instances and setting object and data properties. This is so far done manually. Same goes for the retrieval of information by queries. However, several tools for the automatic instantiation of data from tables or other data sources that use the application programming interface of Protégé exist and could be applied. Additionally, computational knowledge inference by software agents, i.e. issuing queries and receiving and processing the answers, is possible with fully formalized ontologies. Thus, a full automation of the knowledge modeling and reasoning process is considered possible.

## 4.6 Conclusion

The section presented the development, structure and evaluation of a formal ontology for the representation of fixture design knowledge. The ontology serves as integrated information model for the complete fixture design process of the flexible vise and allows for the semi-automatic generation and determination of fixture candidates for a given workholding job by a reasoning engine. Thus, the ontology can be used for fixture configuration design. Fixtures, fixture components and WPs can be represented in a general manner via geometric and functional details modeled as object and data properties.

Through the developed ontological structure and the use of SWRL rules several design constraints can be embedded in the reasoning process to narrow down the selection of fixtures to ones that are geometrically feasible. As not all feasibility criteria can be included in ontological reasoning, a further verification and evaluation of the candidates is required in subsequent FD process steps. The information generated during these steps that is essential for a fixture design, such as a feasible clamping force or the fact whether cutting tool-fixture interferences occurs or not, can be added to the ontological representation of a fixture candidate to generate a formal fixture design model for a specific WP. However, in case no feasible fixture built from the existing fixture components can be determined, the flexible vise supports the generation of new vise jaws by machining. An approach for the automated synthesis of new jaw designs is presented in the next section.

# 5. Computational Synthesis of Fixture Components

As presented in Section 4, the automated fixture design (AFD) ontology can be used to determine fixture configuration designs, also called fixture design (FD) candidates, by reasoning based on the semantic knowledge about the geometric models of the workpiece (WP) and the available jaw designs. This allows for the reuse of existing fixture components, e.g. vise jaws, when possible. In case no feasible FD can be generated using any of the existing components, new component designs need to be synthesized. This is necessary to make use of the capability of the flexible vise to produce new vise jaws from blanks by machining. A spatial grammar, based on a set grammar formulation, is developed for the automation of this design synthesis task. The grammar is called the jaw set grammar as it should generate a set of two jaws for a given WP. While the described grammar is specific to the jaw designs of the flexible vise, the basic approach can be applied for the synthesis of other fixture or manufacturing system components. Hence, the grammar development and underlying ideas are first presented in a rather general manner before they are applied to the specific jaw set design task.

## 5.1 Spatial Grammar Development

To systematically develop the spatial grammar a meta-process is laid out analogous to the ontology development process presented in Section 4.1. The process is depicted in Figure 5-1.

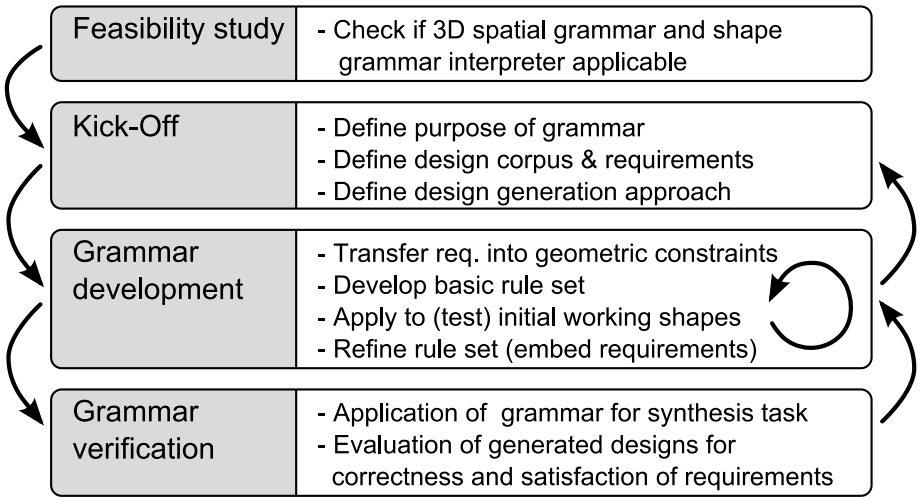



**Figure 5-1 Meta-process for spatial grammar development**

As a first step, the feasibility of using a spatial grammar for the given design task is checked. As no such work has been presented before, the feasibility study needs to be done by developing an initial grammar for the dedicated purpose. An informal, paper-based grammar is sufficient to prove the applicability of a grammar to the design problem. However, automatic grammar application is not possible with a paper-based grammar. As automatic application is a key aspect

in the given scenario, the grammar should be formally implemented to test the expressivity and capabilities of the shape grammar interpreter (SGI) *Spapper* and ensure that it suits the needs.

In the next step, the purpose of the grammar has to be clearly defined. Based on this definition, the requirements for the synthesized designs can be derived. These requirements need to be transformed into design constraints that restrict the possible solution space and define parametric dependencies and allowed parameter values. Further, the basic approach to the design generation has to be defined in this stage. This includes thoughts about whether to use a manual or automated grammar application, an additive or subtractive design generation, how the design constraints are going to be modelled and what the vocabulary and initial working shape (IWS) will be.

The grammar rule set can then be developed in an iterative process based on the results of the kick-off phase. A first rule is generated, taking only some core requirements into account, and tested by applying it to one or more IWSs. In case different IWSs are possible, the test cases should represent the range of possible IWSs. The rule is refined and more requirements are included until it generates the expected results. Then, the next rule can be created and tested. If the rules are sequential, i.e. the second rule requires the application of the first, the designs generated by the first rule are to be used as IWS for the second rule. This process is carried out until a complete basic rule set is generated. The set is then tested using the original IWS. This often identifies refinement needs that require changes to several rules or a re-arrangement of the rules. This loop is continued until all (possible) requirements are included in the rule set. The rule development stage can have an effect on the IWS definition, as data required by the rule set is identified that may have to be included in the IWS as geometric entities, labels or obstacles.

Once the rule set is complete, it can be applied to generate designs for the previously defined purpose. The resulting designs have to be verified for correctness and compliance of the design requirements through visual inspection or using computational analysis methods. This stage can identify the need for changes to the rule set, which causes a return to the rule set development stage.

In the following sections, the steps of the process as carried out for the development of the jaw set grammar are presented. The feasibility study is only briefly covered, as it was presented in a previous publication (GMEINER & SHEA 2013b). Then the kick-off, rule set development and grammar verification are presented. For the rule set development stage, only the resulting rule set is presented, not the iterations. The verification of the grammar is split into a practical evaluation and a discussion of the results.

### 5.1.1 Feasibility Study: A Grammar for Predefined Workpiece Shapes

To validate the applicability of a spatial grammar and the SGI *Spapper* for fixture component synthesis, a first draft of a formal jaw set design grammar was created (GMEINER & SHEA 2011; GMEINER & SHEA 2013b). This represents the feasibility study phase in the presented grammar development process. The initial grammar is built to generate FD candidates for box-shaped and cylindrical WPs, a subset of the WP shapes considered in the approach. Only three design requirements are included in the grammar: 1) the ability of the generated jaw set designs to

provide WP stability and accurate locating, 2) that the jaw design must not affect the functional features of the raw jaws, like those required to align the jaws at the fixture device, and 3) that the generated jaw designs meet the outer dimensions of the given jaw blanks. All these requirements are discussed in Section 5.2.3.

Using the WP model as initial working shape, the initial grammar is able to successfully generate jaw sets for cuboidal and cylindrical WP. This proves the feasibility of the approach and the usability of the SGI *Spapper*. However, the solution space exploration is drastically restricted in this grammar. All design parameters are either fixed or directly depend on a WP dimension. This means that for a given WP model, the resulting jaw set design is always exactly the same. Further, this grammar requires the WP to be either a box or a cylinder. No other shapes can be processed, not even box-shaped or cylindrical WP with additional features. To use grammar-based computational design synthesis for the complete envisioned WP spectrum, a grammar for non-predefined WP shapes is necessary. This grammar must also include more design requirements to be able to produce detailed, production-ready jaw set designs. This requires a different structure of the spatial grammar, as explained in the following section.

## 5.2 Spatial Grammar Kick-Off

In order to develop a grammar that defines a formal language of functional jaw set designs, the design space, design parameters and constraints on the parameters have to be determined. To do so, the purpose and scope are first defined before a corpus of basic jaw set that the grammar should be able to generate variations of designs is determined. Additionally, the design requirements and the available vocabulary provided by the SGI need to be taken into account to define the design parameters and constraints.

### 5.2.1 Purpose and Scope of the Jaw Set Grammar

The goal of using the parametric spatial grammar approach is to automatically generate one or more 3D models of FD candidates by synthesizing new jaw set designs based on a given WP model.

The grammar must be able to generate jaw sets for the accurate locating and clamping of WP that have a generalized polyhedral shape, the shape spectrum defined in Section 1.5. To generate results for a given design problem with a spatial grammar, the possible solution space needs to be defined within the grammar. This is especially important when the grammar is applied automatically, as no user-interaction is wanted to control the design generation. Classically, spatial grammars are used to generate a high number of different conceptual design candidates to explore the solution space. For such a scenario, design emergence plays an important role. In the given scenario, however, the spatial grammar is used for the generation of detailed designs for a constrained engineering design problem. As common in engineering design problems, the solution space is restricted by design requirements that the resulting designs need to meet to ensure functionality. Embedding requirements or, to be precise, the design constraints that result from the requirements, in the grammar, restricts the solution space and, thus, also the variety of the generated solutions. At the same time, it ensures that the generated designs meet the requirements, which reduces the need for subsequent analysis of

constraint satisfaction. As the focus in FD is not on emergence but on the generation of functional designs, including design requirements in the rule set is advantageous.

In comparison to the initial grammar, more design requirements should be included in the final grammar and it should be non-deterministic to allow for an exploration of the solution space. As said, creating widely different designs is not important in FD. However, even slight geometric variations of the designs can have a significant impact on its performance, as will be shown in Section 6. The generation of varying design candidates is, therefore, desirable.

Based on the development of the initial grammar, a set of rules to create jaw sets for the complete WP shape spectrum discussed in Section 1.5 is created. This set of rules substitutes the ones presented in the feasibility study, as it can also process box-shaped and cylindrical WPs. The task of generating jaw set designs for widely different WP shapes raises an issue for the grammar development and makes a simple extension of the existing initial grammar not favorable. The initial grammar uses the WP model as working shape and source of information for the generation of the jaws. The WP model has to be one solid box or cylinder primitive, albeit parametric variations are allowed. The geometric data of the primitive, such as length, height or radius, and the translational and rotational placement parameters are used to define the design variable values of the fixture components. Therefore, the grammar only works if the rules find a match for a (parametric variation of a) box or cylinder primitive in the initial working shape. To boxes or cylinders with additional features, yet alone WPs of another shape, the rules do not apply. To extend the initial grammar to cover the complete WP spectrum, at least one rule for every possible WP shape would be necessary. Considering the nearly infinite WP design variations that the WP spectrum allows, this is next to impossible. Thus, another approach to link the rules to the WP geometry and dimensions is required.

The jaw set only directly interacts with the locating, supporting and clamping surfaces of the WP. The rest of the WP shape has no influence on the required jaw set shape. Thus, to build a jaw set that geometrically fits a given WP, only the shape, dimensions and placement of the contact surfaces need to be explicitly available to the grammar. Instead of using the WP model itself to provide that data, as done in the initial grammar by matching the WP shape, a set of 3D labels that needs to be present in the IWS is used. For every contact surface a label is added to the WP model. Via the placement of the labels, the necessary geometric information is provided. By giving names to the labels, e.g. plane or cylindrical, information about the shape of a surface is provided. Using such a set of labels, the rules do not have to match the WP shape itself anymore but “just” have to match the labels. Thus, the labels simplify the matching process. The structure (presence, naming and placement) of labels required in an IWS depends on the WP shape and is presented in Section 5.3.3. Although this changes the basic approach towards design generation a new feasibility study is not required as the used SGI does support 3D labels.

### 5.2.2 Corpus of Jaw Set Designs

The jaw set grammar should be able to generate jaw set designs that follow a corpus or set of designs that are known to be functional. This corpus is created based on two essential requirements that every machining FD should fulfill: the ability to accurately locate and securely clamp a WP. In FD literature, these requirements are called locating completeness and

WP stability and describe necessary contact and force equilibrium conditions. The adherence of locating and clamping principles and common FD guidelines can ensure that these requirements are met by a FD. For vise-type fixtures, the most common locating principles are 3-2-1 and V-block locating (see Section 2.1.1) and the important design guidelines are (HOFFMAN 2004, p. 21f.):

- In single-acting vises the anvil jaw should serve as the locating jaw, as its position is fixed in space[3].
- The locating jaw should contacts with the WP according to a locating principle.
- The slide jaw should be used for clamping and has to establish contact to the clamping surface of the WP.
- The additional contacts at the slide jaw should not cause an over-determined locating of the WP. However, to ensure locating completeness and stable (un-)loading of the fixture, a second horizontal contact surface at the slide jaw is required, on which the WP can rest when the vise is not engaged.

Based on these principles and guidelines four jaw shapes that can be combined in four ways into jaw sets are defined as design corpus. The jaw shapes are illustrated in Figure 5-2 and the jaw sets in Figure 5-3. The basic jaw shapes presented in Figure 5-2 include step jaws (I), V-block jaws (II) and step-rest jaws. Two different versions of step-rest jaws are possible, one with a contact surface facing in positive y-direction (III) and one with a contact surface facing in negative y-direction (IV).

The use of these four jaw shapes and sets alone does enable but not guarantee locating completeness. For example, if a wrong locating principle is used to locate a WP, locating completeness cannot be achieved. To make sure that a correct jaw set design is generated for a given WP, the knowledge about which WP shape requires which type of locating and clamping jaw shape has to be encoded in the grammar rules. Depending on the shape of the WP not every jaw set design leads to accurate locating and clamping. Step jaws do not allow for accurate locating of a WP as they only provide one horizontal contact surface. Thus, they can only be used for clamping. Step-rest jaws allow for locating a WP following the 3-2-1 principle and V jaws allow for locating a WP following the V-block principle. The selection of the locating principle to be used for a certain WP is solely based on the shape of the secondary locating surface of the WP. Planar secondary locating surfaces require the use of the 3-2-1 locating principle and, thus, a step-rest jaw for locating. Cylindrical primary locating surfaces require the use of the V-block locating principle and, thus, a V jaw for locating. For clamping, step jaws or V jaws can be used. Planar WP clamping surfaces are to be clamped with step jaws, leading to an area contact. Cylindrical WP clamping surfaces can be clamped with step jaws or V jaws, causing either one or two line contacts. Step-rest jaws cannot be used for clamping as they would cause an over-determined locating. Thus, the jaw shapes can be combined into the four different jaw sets shown in Figure 5-3.

[3] The flexible vise is a single-acting vise. Using the fixed anvil jaw for locating avoids issues with a possible inaccurate positioning of the slide and facilitates WP (un-)loading.

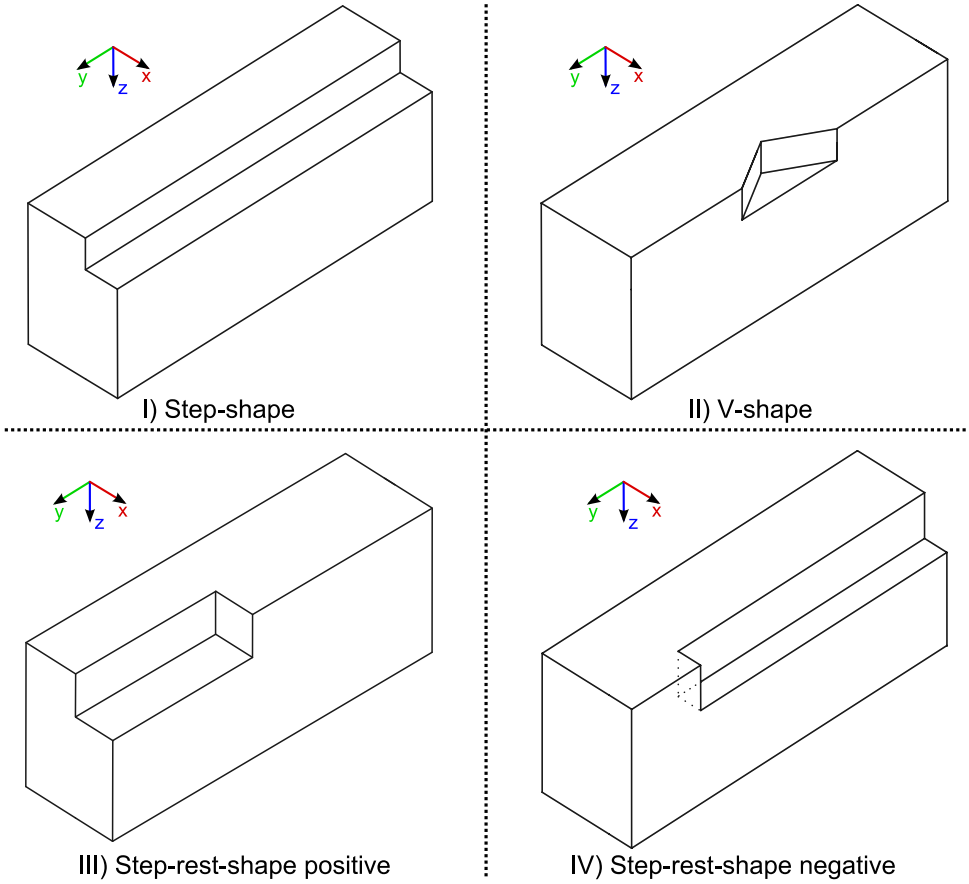



**Figure 5-2 Basic jaw shapes defined as corpus for the jaw set grammar development – I) step-shaped jaw, II) V-shaped jaw, III) step-rest-shaped jaw with y-locating surface in positive y-direction, IV) step-rest-shaped jaw with y-locating surface in negative y-direction**

WPs with a planar secondary locating surface cause the use of a jaw set consisting of a step-rest jaw as locating jaw and a step jaw as clamping jaw. Depending on the orientation of the tertiary WP locating surface either a positive (combination I in Figure 5-3) or a negative (combination II in Figure 5-3) step-rest jaw is used. WPs with a cylindrical secondary locating surface cause the use of either a jaw set consisting of a V jaw as locating jaw and a step jaw as clamping jaw (combination III in Figure 5-3) or a jaw set consisting of a V jaw as locating jaw and another V jaw as clamping jaw (combination IV in Figure 5-3). A combination of a planar secondary locating surface and a cylindrical clamping surface is not allowed, based on the setup guidelines presented in Section 3.8.

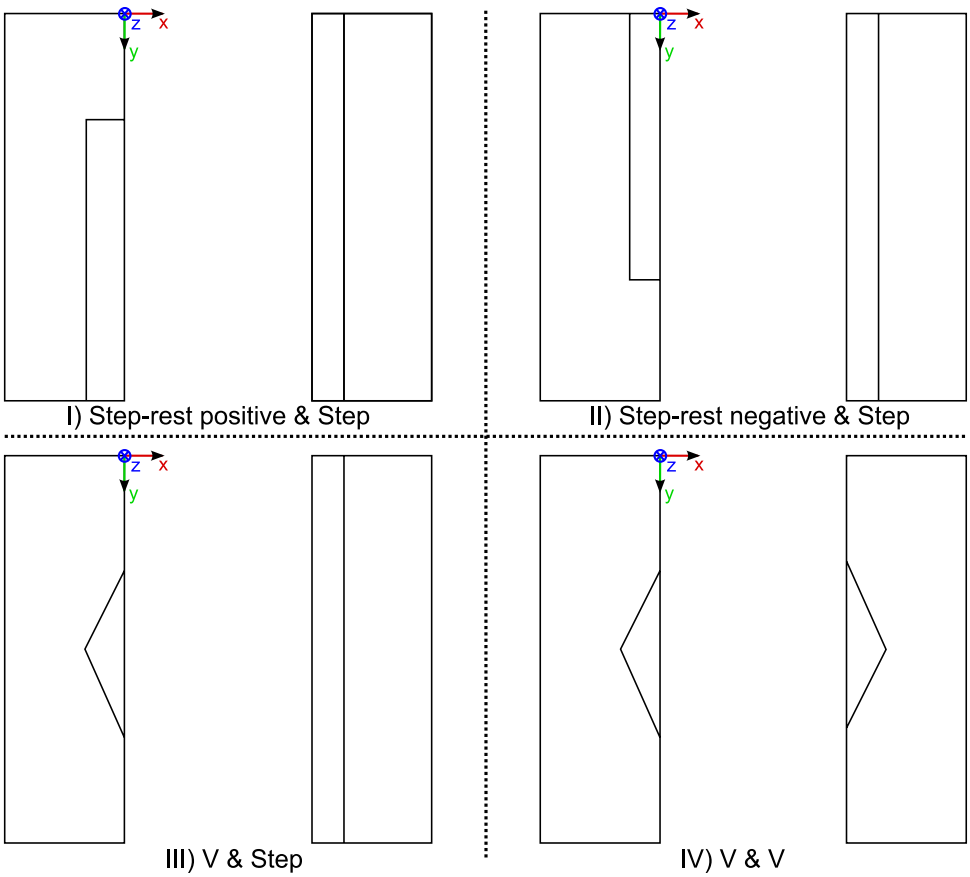



**Figure 5-3 Possible combinations of jaw shapes into jaw sets with left jaw used for locating (anvil jaw) and right jaw used for clamping (slide jaw) – I) positive step-rest jaw with step jaw, II) negative step-rest jaw with step jaw, III) V jaw with step jaw, IV) V jaw with V jaw**

The basic jaw shapes and possible jaw sets are listed in Table 5-1 (left) together with the fixture function and locating principle the jaw shape follows and the shape of the workpieces' secondary locating and clamping surface that the jaw set combination relates to. Additionally, the possible jaw combinations are listed in the combinatory matrix shown on the right side of Table 5-1. The possible combinations of WP locating and clamping surface shapes cover the complete WP spectrum considered in this work. Therefore, all possible WP shapes can be located and clamped with one of these jaw sets, given that the WP is correctly set up and that the detailed design of the jaw set meets the actual fixture problem.

The presented basic jaw set designs represent the corpus of designs that the grammar should be able to generate variations of. As common in engineering design problems, the range of variations that is feasible for a given WP is subject to the satisfaction of design requirements. This is discussed in the next section.

**Table 5-1 Possible basic jaw shapes and jaw sets together with the locating principle and the shapes of the workpiece contact surfaces a jaw set relates to (left) plus combinatory matrix of basic jaw shapes into sets (right)**

| Jaw shape | Fixture function | Locating principle | Possible jaw sets | WP surface shapes | |
|---|---|---|---|---|---|
| | | | | Secondary loc. | Clamping |
| Step-rest pos. | Locating | 3-2-1 | Step-rest pos. & Step | Planar | Planar |
| Step-rest neg. | Locating | 3-2-1 | Step-rest neg. & Step | Planar | Planar |
| V | Locating | V-block | V & Step | Cylindrical | Planar/Cyl. |
| | Clamping | -- | V & V | Cylindrical | Cylindrical |
| Step | Clamping | -- | Step-rest & Step | see above | see above |
| | Clamping | -- | V & Step | see above | see above |

| Slide \ Anvil | Step-rest | V | Step |
|---|---|---|---|
| **Step-rest** | 0 | -- | -- |
| **V** | 0 | 1 | -- |
| **Step** | 1 | 1 | 0 |
| 0 = no combination<br>1 = combination | | | |

## 5.2.3 Jaw Set Design Requirements

In an engineering design problem, the solution space is generally restricted by design requirements. Thus, these design requirements have to be embedded in the grammar to generate valid designs. In FD, the requirements stem from the geometric, kinematic and functional constraints imposed by the fixture device, its environment, such as the machine tool or the manufacturing system, and general fixture design guidelines, such as locating principles. Examples of such requirements are the maximum clamping width of a vise or the maximum dimensions of the blanks that fixture components can be machined from. Further, requirements that represent design feasibility are an important issue. These requirements can be the generation of interference-free designs or the adherence of locating accuracy constraints, depending on what the key aspects of design feasibility for a user are. To work with the requirements, they are made explicit by creating a list of requirements. The list of requirements for the flexible vises' jaw set designs is given in Table 5-2. The requirements can be either fixed, meaning that they must be satisfied, or flexible, meaning that they should be satisfied. To keep it concise, the requirements are only briefly presented. A more detailed discussion is given in Appendix 10.6.1.

The first and second requirement imply that the generated jaw sets must accurately locate the WP, in a well-determined manner, and establish form or force closure, such that the WP does not move when it is subject to the machining forces. The third requirement states that the generated jaw designs must be machinable from the given jaw blanks without altering the functional features. The functional features, as shown by the dashed lines in Figure 5-5, are on the backside and on the top of the raw jaws and are used for handling of the jaws by the robot grippers and for establishing the temporary joint to the anvil and slide of the vise. The given dimensions of the jaws and these features have an influence on the volume that can be removed from the jaw to create the contact features by machining.

Next a few functional requirements (4-8) are listed which state that the generated jaw set designs must be free of tool-fixture collisions, allow for stable loading and unloading of the vise by the handling robot, that the designs must be machinable with the given machine, cutting tools and jaw machining process (jaws pressed against each other for machining as shown in Figure 1-3) and that the WP can have two different horizontal contact surfaces. The rest are geometric restrictions that are imposed by the fixture device, such as the maximum opening length, or defined by the user, such as the minimum remaining material thickness requirements.

**Table 5-2 List of requirements for the development of the jaw set grammar that must or should (fixed/flexible) be embedded in the designs synthesized by the grammar**

| **Gmeiner** | | **List of Requirements<br>Development of jaw set grammar** | **Page 1/1** | |
|---|---|---|---|---|
| **Nr.** | **Fixed/<br>Flexible** | **Requirement** | **Value** | **Unit** |
| 1 | Fix | Accurate locating for generalized polyhedral WP shapes | -- | -- |
| 2 | Fix | Ensure WP stability during machining | -- | -- |
| 3 | Fix | No alteration of functional features of jaws | -- | -- |
| 4 | Fix | No tool-fixture interferences | -- | -- |
| 5 | Fix | Allow for stable (un-)loading of vise | -- | -- |
| 6 | Fix | Ensure machinability of designs with existing cutting tools | -- | -- |
| 7 | Fix | Ensure machinability of designs with machining process | -- | -- |
| 8 | Fix | Different supporting surfaces at anvil and slide possible | -- | -- |
| 9 | Fix | Compliance with given raw jaw model and dimensions | 28x100x40 | mm |
| 10 | Flex | Maximum opening length restriction of vise | 120 | mm |
| 11 | Fix | Minimum opening length restriction of vise | 4 | mm |
| 12 | Flex | Minimum height for locating/clamping surfaces of jaws | 2 | mm |
| 13 | Flex | Opening angel of V-blocks | 120°-140° | deg |
| 14 | Flex | Minimum remaining material thickness in x-direction | 2 | mm |
| 15 | Flex | Minimum remaining material thickness in y-direction | 10 | mm |
| 16 | Flex | Minimum remaining material thickness in z -direction | 5 | mm |

While all requirements have an influence on the feasibility of a final jaw set design, not all of them may be embeddable in a spatial grammar. As spatial grammars only work with spatial

objects, the requirements need to be converted into constraints on the parameters that are included in the grammar vocabulary or rules. For geometric requirements, this process is rather straight forward. However, for higher-level functional requirements, like to "ensure machinability", this is a non-trivial task. If requirements cannot be embedded in the grammar, subsequent design verification with respect to these requirements becomes necessary.

### 5.2.4 Approach to Automatic Grammar Application

A core aspect of the jaw set grammar is the ability to automatically apply it. This includes automatic rule selection, LHS matching and setting of parameter values. The used spatial grammar interpreter *Spapper* allows for automatic grammar application by supporting parametric left-hand side (LHS) matching including Euclidean placement transformations, sequential or random (i.e. non-sequential) application of rules and generation of multiple solutions. The SGI further allows for the random setting of parameter values. For the automatic application to work, not only the respective settings in the SGI have to be activated, but the grammar rules have to be modeled accordingly.

As will be shown, random rule selection is necessary for the non-deterministic generation of different combinations of basic jaw designs. Further, it facilitates the use of the rule set as it reduces error-proneness to wrong rule sequence definitions. To still allow for the generation of correct designs in random rule selection mode, the applicability of the rules is restricted to certain states in the design synthesis process using state labels. As explained in Section 2.3.1, this implies that one or more labels are used on the LHS of a rule to restrict its applicability to states where these labels are present in the current working shape. Once a rule is successfully applied, it changes the state such that one or more other rules become applicable. State changes are also used for the automatic termination of the rule application process. An application of a final rule changes the state such that no more matches for any of the rules can be found. When no more applicable rules can be found, the system automatically ends the synthesis process.

To enable an automatic setting of parameter values, the parameters of the objects (shapes or labels) have to be defined as 'free parameters' during the rule development. Free parameters can be either set unrestricted, which allows for a completely free selection of values by the SGI, or a minimum-maximum range for the parameter can be given or they can be defined via equations. This is done in the parameter definition window of the SGI that is available for each object, both on the LHS and right-hand side (RHS) of a rule. Free parameters on the LHS allow for matching of objects that parametrically vary in dimensions or placement. Free parameters on the RHS allow for the generation of such objects. The parameters include the dimensions of the object, such as height, length and width, and its placement in 3D space represented via three translational (TranslateX, TranslateY, TranslateZ) and three rotational parameters (Yaw, Pitch, Roll). Free parameters can be used in the definition of parameters to model dependencies. Further, mathematical operations can be included in the parameter definitions.

The basic jaw shapes and jaw set definitions as well as the design requirements are the basis for the development of the formal jaw set grammar. This formalization is described in the next section.

## 5.3 Development of the Jaw Set Grammar

To enable the synthesis of design candidates with the grammar, the basic designs need to be expressed using the available vocabulary provided by the SGI. The vocabulary directly defines the existing design parameters. The design requirements and basic jaw set designs define which of the parameters are fixed and which are variable as well as interdependencies among the parameters and value ranges for the variables. These constraints then need to be modeled as parameter definitions in the grammar rules. Considering the development meta-process presented in Figure 5-1, Sections 5.3.1 to 5.3.3 represent the stage of defining the design generation approach. Section 5.3.4 then presents the final rule set development stage without explicitly discussing the iterations that were required for the development.

### 5.3.1 Grammar Vocabulary and Design Parameters

*Spapper*, the SGI used for the development and application of the jaw set grammar, limits the available vocabulary to build the grammar from solid primitives and objects created through Boolean operations with solid primitives or sweeping operations with 2D sketches. Applying this to the basic jaw shapes presented in Figure 5-2 yields the parametric jaw shapes shown in Figure 5-4.

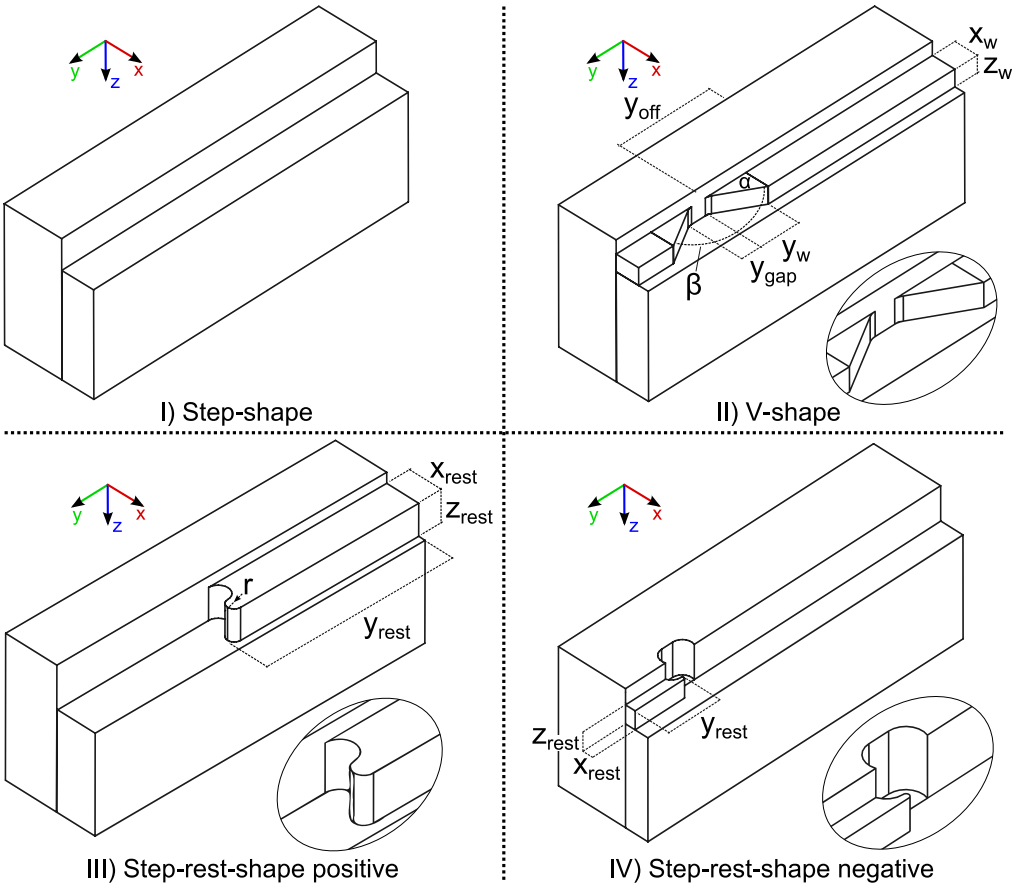



**Figure 5-4 Jaw designs built with available grammar vocabulary with close-ups of cutouts and fillets – I) step-shaped jaw, II) V-shaped jaw, III) positive step-rest-shaped jaw, IV) negative step-rest-shaped jaw**

The jaw designs are slightly altered to allow for more design variations and to ensure the machinability of the designs. All jaws are built using several instances of the solid primitives box and wedge. Further, objects built from Boolean union and intersection operations are used to generate the rounded pocket and fillet features, which are added to the basic jaw shapes to ensure the machinability of the designs with a 3-axis machine tool with vertical tool approach direction, which is the type of machine the flexible vise is installed on (see Section 1.4). The necessity of these features is based on a design requirement, and explained in detail in Section 5.3.4.

The design parameters of the jaws are the dimensional and placement parameters of the primitives used to build the jaws, as depicted in Figure 5-4. The values that the design parameters can or must have, are not arbitrary but restricted by the basic jaw shapes (e.g. that certain surfaces of boxes need to be in touch) as well as by constraints imposed by the design requirements. Some fixed constraints on dimensional parameter values imposed by requirements and valid for all jaw shapes are illustrated in Figure 5-5. The figure shows a blank jaw with the functional features on the backside and top of the jaw in dashed lines and the volume that can be used to machine the contact features of the locating and clamping jaws highlighted in grey. The given raw jaw model defines the maximum outer dimensions of the jaws. From this box material must be removed by milling operations to create the necessary locating, supporting and clamping surfaces. The functional features, the user-defined minimum remaining thickness limits and the maximum length of the smallest cutting tool restrict the maximum cutting depth ($z_{step,max}$ in Figure 5-5) and maximum cutting length ($x_{step,max}$ in Figure 5-5).

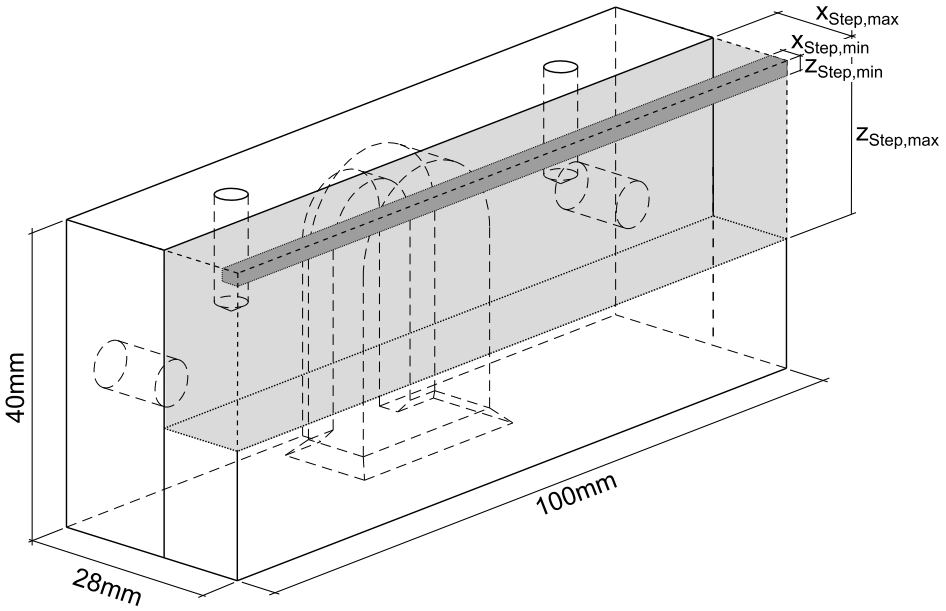



**Figure 5-5 Raw jaw with functional features (dashed lines) and minimum (dark grey) and maximum volume (light grey) to be removed by machining operations to create the contact surfaces**

The minimum cutting depth ($z_{step,min}$) is defined by the requirement for a minimum height of the locating and clamping surfaces. The minimum cutting length ($x_{step,min}$) is defined by the machining process of the jaws and the minimum cutting tool diameter. With the given fixture device, the jaws have to be pressed against each other to enable their machining. This is primarily necessary to accurately position and to stabilize the slide jaw. The minimum cutting length can be achieved when the centerline of the cutting tool is positioned in the contact plane of the jaws, as shown on the left side of Figure 5-6. The cutting length at both the anvil and the slide jaw then equals half of the tool diameter. If the WP has different supporting surface heights on the anvil and slide jaw, the minimum cutting length of the jaw with the deeper cutout equals the whole minimum tool diameter, as shown on the right side of Figure 5-6. Besides the minimum value, the cutting lengths at both jaws are independent from each other. The detailed values for the removable volume of the flexible vise jaws are presented in Appendix 10.6.2.

In the jaw set designs generated by the grammar, the functional features (see Figure 5-5), which are already present in the raw jaws, are not modeled. This facilitates the grammar and is feasible, as for the subsequent machining of the jaws only the features added to the raw jaws are important, as only these have to be machined.

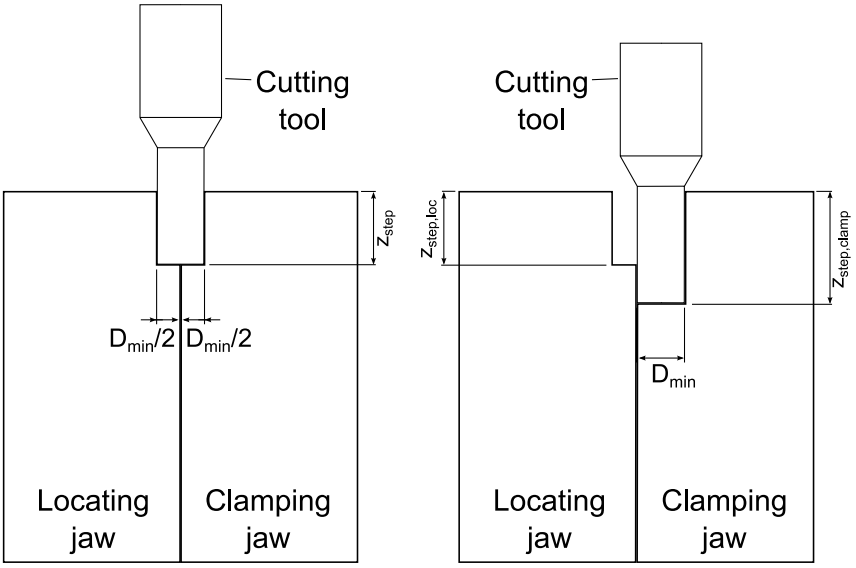



**Figure 5-6 Dependence of minimum step length from cutting tool diameter ($D_{min}$) in case of equal (left) or different (right) supporting surface heights at locating and clamping jaw shown in side view**

## 5.3.2 Obstacles for the Generation of Interference-Free Designs

Another aspect that is discussed preliminary to the actual rule set is the inclusion of the requirement that the jaw set designs must be free of tool-fixture interferences. This is realized using spatial obstacles, a feature provided by the SGI *Spapper*. An obstacle is a 3D object that restricts the design space, i.e. the grammar cannot generate geometric objects that collide with an obstacle.

By adding an obstacle to the WP model that represents the volume that the cutting tools require for the machining of the WP, the requirement for interference-free designs can be embedded in the grammar. This volume is called a tool swept volume (TSV), as it can be created by sweeping the cutting tool models along the machining paths. The approach of automatically generating a TSV and adding it to the WP model is presented in Section 6.1. In the generation of an initial working shape (see next section) this solid is added to the WP model and defined as an obstacle. By using the TSV as an obstacle in the design synthesis process, all jaw set designs generated are guaranteed to be free of tool-fixture interferences as the built jaws are not allowed to touch the obstacle.

### 5.3.3 Preparation of the Workpiece as Initial Working Shape

As described in Section 5.2.1, the jaw set generation has to be based on the WP model. To circumvent the LHS matching problem of (at the time of the grammar development) non-predefined WP shapes, a set of labels is used instead of the WP model itself, to provide the necessary geometric data for the jaw set synthesis. This set of labels has to be manually added to the WP model to allow for its use as initial working shape.

Throughout the process of rule application, the labels work as state and spatial labels, as mentioned in Section 3.5. The labels mark the locating, supporting and clamping surfaces and the center-of-mass (CoM) of the WP. For each setup surface a label has to be added. The label names, such as *Loc_2* or *Loc_2_cyl*, define whether the WP surface is planar or cylindrical. Within the SGI *Spapper*, each label has three translational and three rotational parameters but no dimensions. By setting the parameter values according to the WP surface positions, the geometric data required for the jaw set generation can be provided. For each label only a few parameters need to be set. For the supporting surface labels, for example, only z-translation parameter needs to be defined, as for the jaw set generation only this information is important. The remaining parameter values do not have to be considered or can be chosen arbitrarily. By having each label store multiple placement values, e.g. one label to store the x-position of the secondary locating surface, the y-position of the tertiary locating surface and the z-position of the primary locating surface, it would be possible to use fewer labels. However, to facilitate a possible use of the placement data in subsequent steps and applications, the separation into individual labels is chosen at this point. Table 5-3 summarizes the necessary labels in the IWS depending on the WP surface shape, the fixture function of the surface (locate, clamp, support) and the placement parameters that have to be set. Note that *Loc_1* must be present only once and that depending on the WP shape, either *Loc_2* or *Loc_2_cyl* is used, but never both in the same IWS.

Two examples of IWS, e.g. fully labeled and oriented WP, are depicted in Figure 5-7. For the sake of clarity, not all labels are shown in each view, although the combination of both views (side and top) includes all labels. Also, the tool swept volume obstacles are not shown in the figure. The WP on the left is setup such that it has cylindrical locating and clamping surfaces. This model requires the labels *Loc_1*, *Loc_2_cyl*, *Clamp*, *Clamp_cyl*, *Support_1, Support_2* and *CoM*. To establish a correct setup, according to the conditions presented in Section 3.8, the WP is oriented such that the *Loc_1* and *Clamp* labels are in one straight line parallel to the x-axis, as highlighted by the dashed line in the top view (bottom left). This is necessary to not

induce a rotating momentum around the z-axis by the clamping force. The WP on the right side of Figure 5-7 is setup such that it has plane locating and clamping surfaces. This requires the labels *Loc_1*, *Loc_2*, *Clamp*, *Support_1, Support_2* and *CoM*. It can be seen that both *Support* labels are at the same surface. Still, the grammar requires the presence of both labels, even if they have the same z-value. In both examples the *CoM* labels cannot be seen as they are inside the solid.

**Table 5-3 3D labels added to the workpiece models/initial working shapes with the WP surface the label relates to and the translational parameters that need to be set to store the geometric information of the workpiece surface (in the vise COS)**

| Label name | WP surface | Set parameters - information stored |
|---|---|---|
| Loc_1<br>OR | Plane secondary locating surface | TransX - x-position of surface<br>TransY - y-position of corner to secondary locating surface |
| Loc_1 | Cylindrical secondary locating surface | TransX - x-position of surface<br>TransY - y-position of center of radius |
| Loc_2<br>OR | Plane tertiary locating surface | TransX - Maximum x-position of surface<br>TransY - y-position at max. x-position<br>Yaw - outward normal direction of surface |
| Loc_2_cyl | Cylindrical locating surface (same as seconary) | TransY - Radius of surface via y-distance to label Loc_1 |
| Support_1 | Plane primary locating surface (at anvil side) | TransZ – z-position of surface |
| Support_2 | Plane supporting surface (at slide side) | TransZ - z-position of surface |
| Clamp | Plane or cylindrical clamping surface | TransX - x-position of surface |
| Clamp_cyl | Cylindrical clamping surface | TransY - Radius of surface via y-distance to label Clamp |
| CoM | None | TransX – x-position of center of mass<br>TransY – y-position of center of mass<br>Yaw – y-axis points towards plane sec. loc. surface (0°/180°) |

The preparation of the initial working shape can be carried out as one process with the setup planning in *FreeCAD/Spapper*. A WP is first oriented according to the setup conditions presented in Section 3.8 and then labeled, taking the label naming and placement conventions presented in Table 5-3 into account. If the model is extended with a TSV, this volume has to be

defined as obstacle using the corresponding function of *Spapper*. The preparation is a linear step-by-step process and the same for all WPs. The algorithm used for this process is presented in Appendix 10.5. An implementation of this algorithm in the CAD environment that walks the user through the steps of selecting the setup entities of the WP, restricts the selection to feasible alternatives and automates the orientation and label generation based on the selection, could be used for interactive setup planning and IWS preparation. Such an implementation is, however, not covered in the presented work.

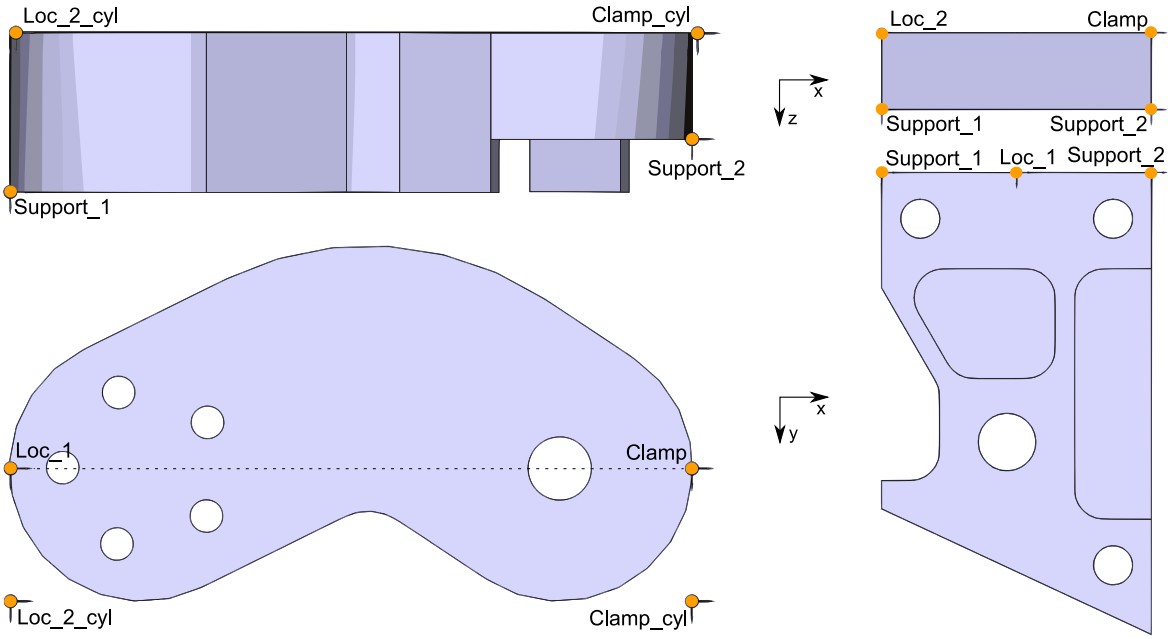



**Figure 5-7 Workpiece with cylindrical (left) and with plane (right) contact surfaces labeled and oriented according to the initial working shape preparation process (superposing labels not shown)**

## 5.3.4 The Jaw Set Grammar

Starting with an IWS created this way, the grammar has to build the jaws around the WP model in an additive manner, i.e. the rules have to create and add solid primitives to the working shape to build the jaw design. The result of the grammar application is a model of the jaw set with the WP in the clamping position. Based on the presented preliminaries, a set of 12 rules for the generation of jaw set designs for the given WP spectrum is developed.

In the following, the rules are described based on their LHS and RHS and their effect on the current working shape, i.e. their effect on the jaw set design process. Detailed descriptions of the parametric definitions and the link to the design constraints would exceed the scope of this documentation and are, thus, not presented here. In Appendix 10.6.2 further details on the case-specific geometric constraints embedded in the parameters can be found. A more detailed discussion of the grammar rules and the underlying design rationale is given in

Appendix 10.6.3. For a detailed study of the parameter definitions the grammar is also provided online (HOISL 2014).

The first rule (R00-X) looks for a starting set of labels in the IWS and, based on the information the labels provide, builds all contact primitives of both the locating and the clamping jaw on the RHS. Additionally, the rules delete or change the labels in the working shape to the state required by the next rule. As four basic jaw set shapes exist, four different cases of the first rule are necessary. Figure 5-8 shows a graphical representation of the four R00-X rules with the LHS and the RHS of each rule. The RHS is also shown in top and side view wireframe for better clarity. The contact primitives that the rules build are two L-shaped bodies, build from two flat boxes, and either another box or two or four wedges. The L-shapes represent the step-cutouts for both the locating and the clamping jaw and have the width of the jaw blanks. An additional box placed inside the corner of one L-shape models a step-rest-jaw and two wedges model a V-jaw. Four wedges occur if both the locating and the clamping jaw are V-shaped.

Once a R00-X rule is successfully applied, rule R01 becomes applicable. This rule, as shown in Figure 5-9, matches a previously created state label *Support* (in yellow) and one L-shape in close proximity to the *Support* label on the LHS. When applied, it extends the L-shape to create a step-shaped jaw with the height and length of the jaw blanks. Including this in the R00-X rules is not possible, as the fixed height or length of the L-shapes, which is 1 mm, is required for the parameter definitions. As two L-shapes and two matching state labels exist in the working shape after the application of R00-X, the rule R01 can be applied twice, once for the locating and once for the clamping jaw. When applied, the rule changes the color of the matched *Support* label to green. With this state label, five different rules are applicable, depending on the shapes created in the R00-X rule.

Rule R02, shown in Figure 5-9, applies if a green *Support* label and two wedge primitives on top of and aligned with a box exist. Thus, it is only applicable after R01 was applied and only if a V-shaped jaw is present. The rule creates add-on boxes on the outside of wedges. The added boxes have the same height and length as the wedges and extend from the outside of the wedge to the respective outside of the jaw. Though these add-ons are not necessary for the function of the jaws, they increase the structural stability of the design and reduce the volume that has to be machined. The dimensions of the primitives on the LHS can vary but to make sure only a correct set of primitives is matched, their placement in regard to the *Support* label and in regard to each other is restricted. Just like in R01, the parameter ranges are defined in a way that allows for a matching of the rule on both the anvil and the slide side. This way, the rule can be either applied once, in case of a V & Step jaw set, or twice, in case of a V & V jaw set. When applied, the rule changes the color of the *Support* label to blue.

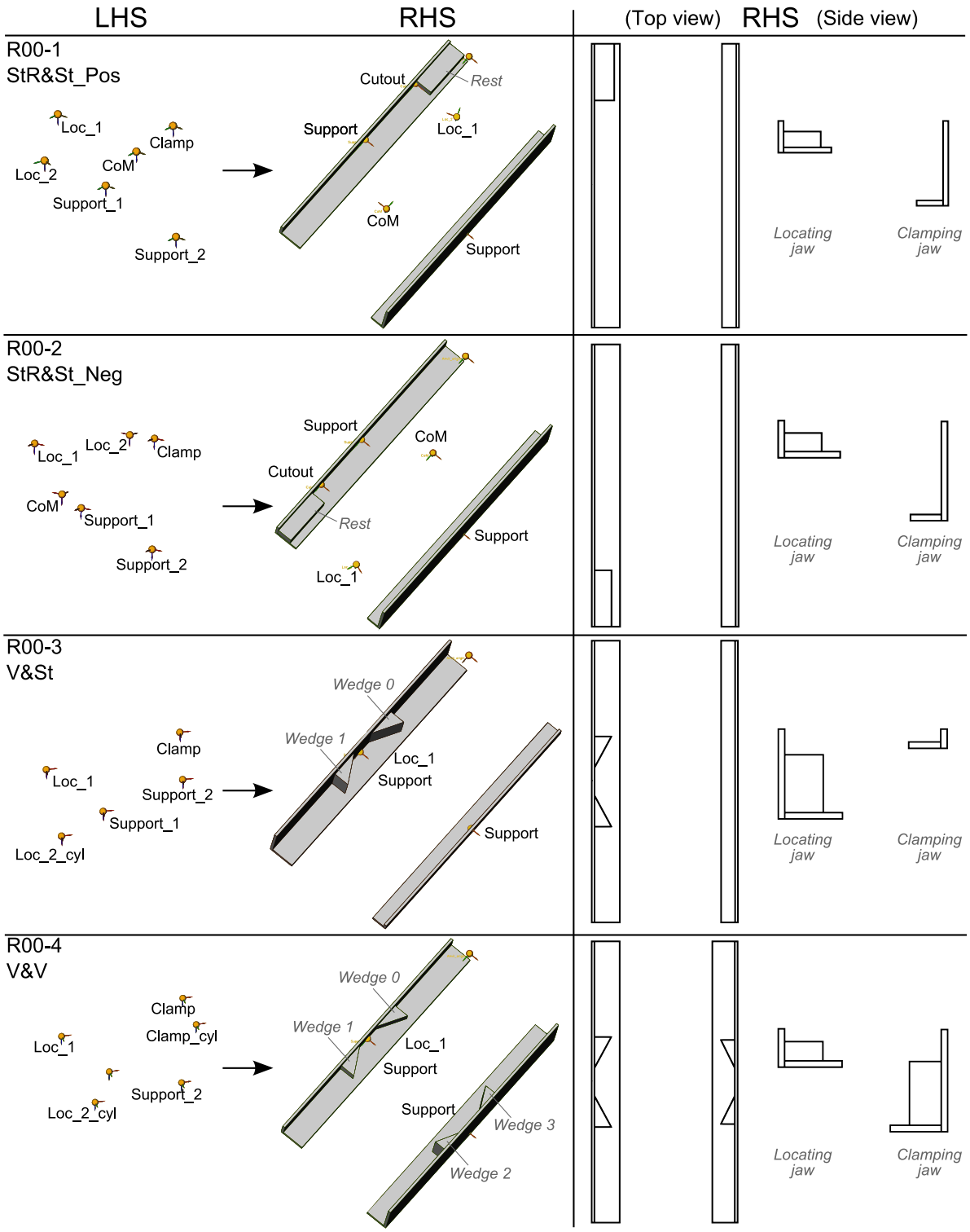



**Figure 5-8 LHS and RHS of the four R00-X rules (left); wireframe model of RHS shown in top and side view (right)**

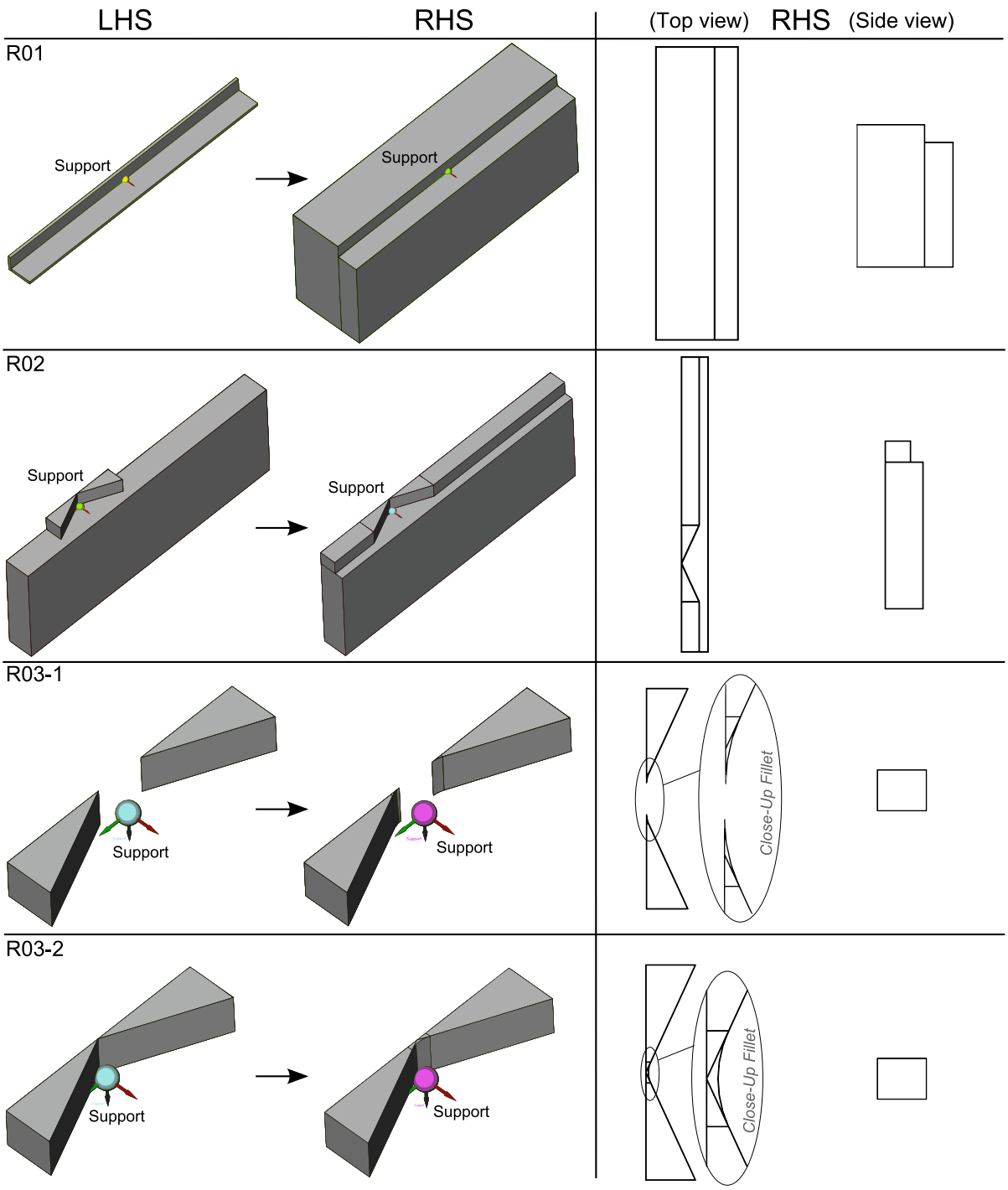



**Figure 5-9 LHS and RHS of the rules R01. R02, R03-1 and R03-2 (left); wireframe model of RHS shown in top and side view with close-ups (right)**

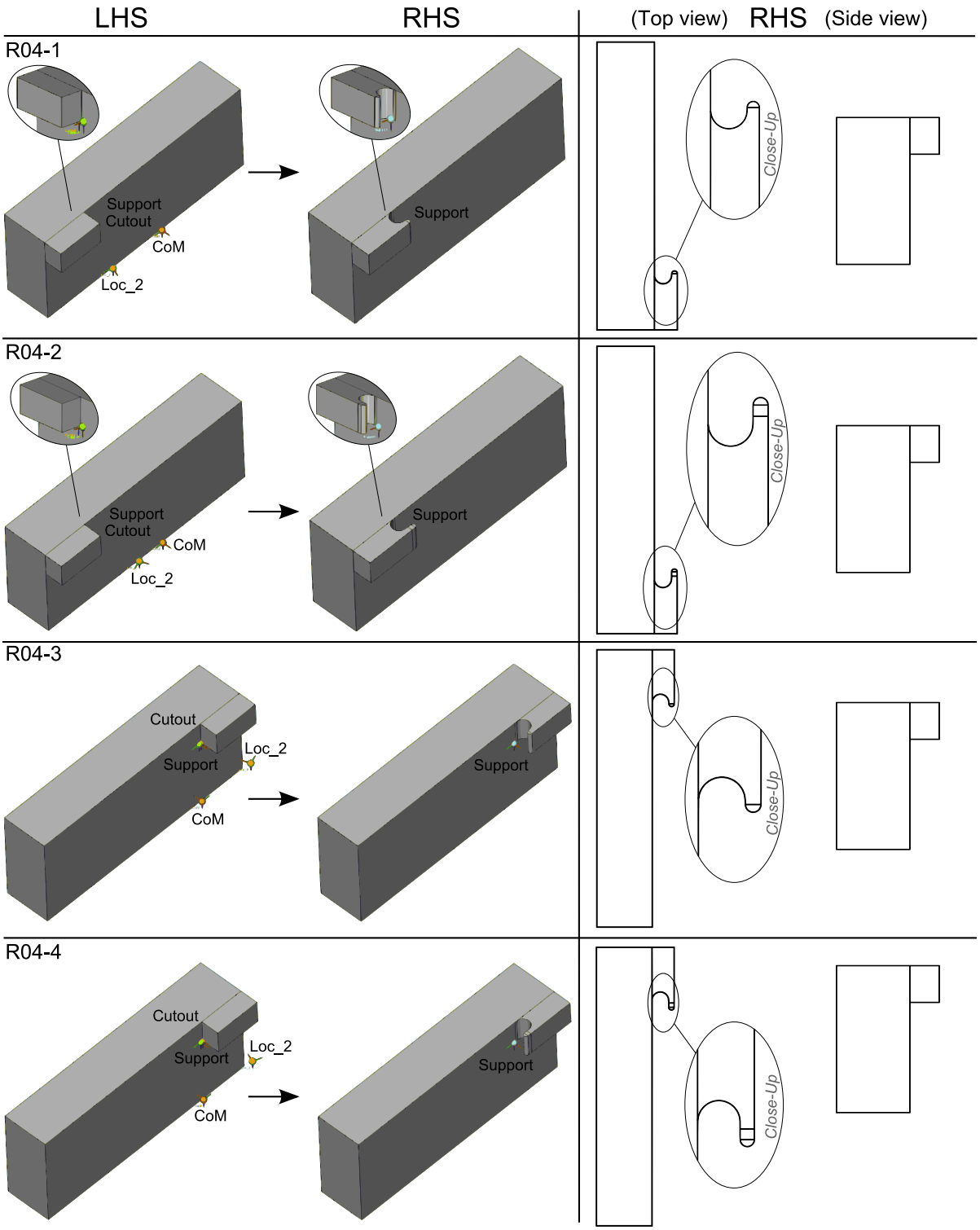



**Figure 5-10 LHS and RHS of the rules R04-X (left); wireframe model of RHS shown in top and side view with close-ups (right)**

The application of rule R02 and the caused change of the *Support* label color to blue is the prerequisite for the application of rule R03-X. Two versions of this rule exist, R03-1 and R03-2, as illustrated in Figure 5-9. They both create a fillet of the inside corner(s) created by the tips of the wedges. These rules alter the shape of a V-jaw such that it can be machined through vertical milling. A differentiation of the rule is necessary as two different approaches to model the fillets are required, depending on the size of the gap between the wedges. If the gap is small enough for the cutting tool to touch both sloped faces of the wedges simultaneously, rule R03-2 can be applied. When a certain width of the gap is exceeded, depending on the angle of the wedges and the cutting tool diameter, this is no longer possible. Then rule R03-1 can be applied. Rule R03-2 adds one solid, and rule R03-1 two, created from the Boolean intersection of a box and a cylinder, to model the fillet(s). Depending on the number of existing V-shaped jaws in the set, the R03-X rules can be applied once or twice. When applied, the color of the *Support* label is changed to pink. This terminates the rule application process as no rules within the rule set match this state label color. The labels are not deleted as they are required in subsequent FD process steps, as explained in Section 6.1.1.

As mentioned before, five rules are applicable when a green *Support* label is present. R02, which applies when a V-jaw exists, was already covered. The other four rules are the R04-x rules, shown in Figure 5-10, which apply in case a step-rest-jaw exists in the working shape. R04-1 and R04-2 apply to step-rest jaws with a negative y-locating direction while R04-3 and R04-4 apply to step-rest jaws with positive y-locating direction. Just like the R03-X rules, these rules alter the existing jaw shape to ensure the machinability of the design by creating a rounded cutout at the inside corner. Further, they round off the secondary locating surface of the jaw. Through this rounding, a line contact is established between the WP and the jaw for locating in y-direction, which allows for the accurate positioning of WP where the primary and secondary locating surface meet at an obtuse, right or acute angle. A surface contact, on the other hand, would only allow for the accurate positioning of a right-angled surface pair. A case differentiation of this rule is required for negative (R04-1 & R04-2) and positive (R04-3 & R04-4) step-rest jaws, due to the placement of the Boolean objects used to create the cutouts. A further case differentiation for each step-rest jaw type is required to deal with different angles between the secondary and primary WP locating surface. Depending on the angle, the rounded contact element of the jaw needs to be translated or extended in y-direction to assure the contact to the secondary WP locating surface is maintained. This is illustrated in Figure 5-11. The yaw angle of the *Loc_2* label is used to provide the information about the surface angle. R04-1 and R04-3 apply to WPs with obtuse angles and R04_2 and R04_4 to WPs with right or sharp angles. If any of the R04-X rules is applied the color of the *Support* label is changed from green to blue. This is the termination criterion for the generation of a step-rest & step jaw set. Although rules R03-1 and R03-2 match blue *Support* labels, they also require the presence of two wedges, which do not exist in a step-rest & step jaw set.

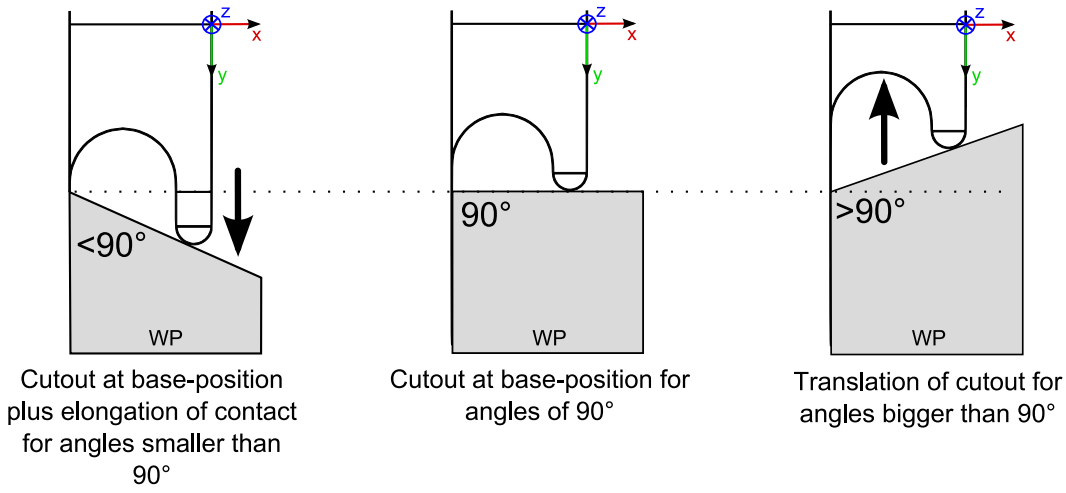



**Figure 5-11 Effect of angle between primary and secondary workpiece locating surfaces on the geometry and placement of the cutout and fillet required to make the inside corner machinable**

An example of a jaw set synthesis process is given in Figure 5-12. The process starts with the IWS of the demo WP Rec_FR in the second of two setups (Rec_FR_1 from Appendix 10.2). The model of the WP is fully labeled, according to the IWS preparation process described before, and is extended with a total TSV which is displayed as semi-transparent red volume. For better visibility, the total TSV is not shown in the intermediate process steps and the WP is depicted as wireframe model. The set of labels, especially the presence of the *Loc_2_Cyl* and *Clamp_Cyl* labels, in the IWS only allows for the application of either rule R00-3 or R00-4. Every IWS that leads to a match of R00-4 also leads to a match of R00-3, but not vice versa. This is due to the fact that the complete set of labels in the LHS of R00-3 is also part of the LHS of R00-4, which is just extended by a *Clamp_Cyl* label.

In the example in Figure 5-12, rule R00-3 is applied as first rule. This builds all contact primitives and defines the basic jaw shape, in this case a combination of V jaw for locating and step jaw for clamping. Due to the random selection of many parameter values, the application of this rule can be subject to collisions with the TSV. Thus, several (unsuccessful) rule applications can be necessary until the rule is successfully applied, i.e. a setting for all parameter values is found that does not cause a collision of the built primitives with the TSV. When applied successfully, the state of the labels is changed. Most labels of the IWS are deleted and two yellow *Support* labels, one for each jaw, are generated. This state only allows for the application of rule R01, which builds the two main boxes of a jaw. This can either be the locating or the clamping jaw, as the current working shape leads to two matches of the LHS of rule R01. The system randomly selects the locating jaw as match. The application of the rule, again, causes a state change: the matched label *Support* at the locating jaw turns green. This current working shape allows for the application of either rule R01, for the clamping jaw, or for the application of rule R02 for the locating jaw. The system randomly selects rule R02, which generates the add-on boxes to the wedges and changes the color of the *Support* label to blue.

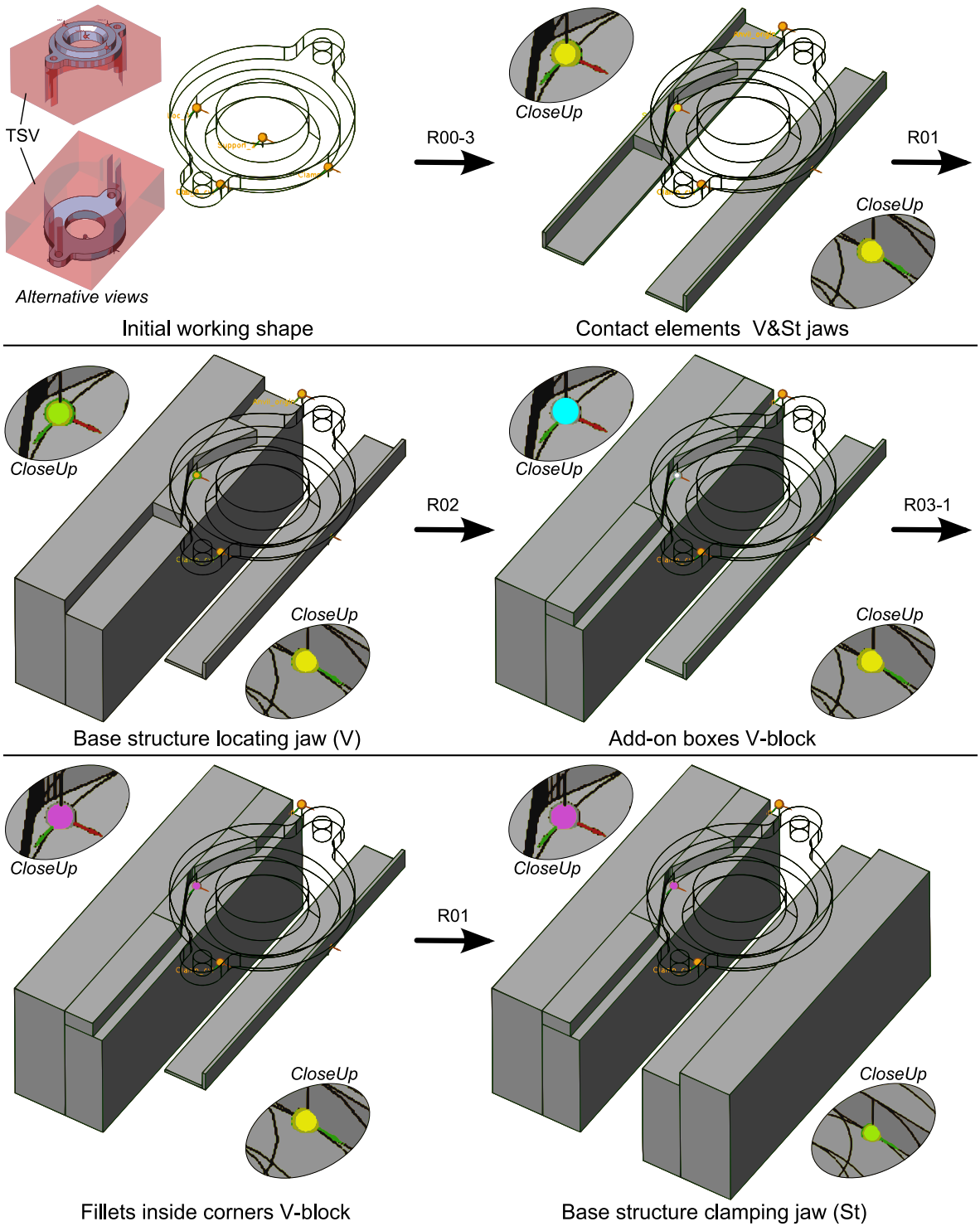



**Figure 5-12 Design synthesis process with workpiece as wireframe model and close-ups of state labels for workpiece Rec_FR_1**

Now, rule R01 for the clamping jaw or rule R03-X for the locating jaw can be applied. The system selects rule R03-1. Based on the gap width between the wedges, only this R03-X rule leads to a LHS match. The application generates a fillet at the wedge tips and changes the color of the matched *Support* label to pink. For this current working shape only rule R01 leads to a match. The application generates the step-shaped clamping jaw and changes the color of the matched *Support* label from yellow to green. This terminates the jaw set design generation process, as the state of the working shape leads to no further matches for any of the rules.

The example shows that the R00-X rules are partially non-deterministic, meaning that for some IWS different R00-X rules are applicable. Each R00-X rule includes several free parameters on the RHS that allow for the generation of parametric design variations. All other rules in the rule set, however, are deterministic, meaning that the first rule already defines the design of the generated jaw set. The generation of all contact primitives of both jaws in the first rule is of advantage to keep the rule set slim. This is due to the many interdependencies of the design variables within each jaw shape and among the jaw shapes of a jaw set. Using a sequential build process of the contact primitives is possible but would result in a much higher number of rules without gaining any advantages, e.g. in terms of design emergence and non-deterministic design generation. Such a rule set would simplify the RHS of the rules but would make the LHS matching more complicated, as all parameters involved in an interdependent parameter definition would need to be present in the LHS of a rule. Thus, the parallel generation of all contact primitives in the first rule was chosen.

## 5.4 Evaluation of the Spatial Grammar

To verify the developed spatial grammar, its capabilities to automatically generate multiple different jaw set designs for all demo WP presented in Appendix 10.2 are tested. This requires the manual planning of setups and the preparation of the WP models, including orientation and labeling of the models. The setups need to be planned according to the setup rules presented in Section 3.8. The setup plan for every demo WP from blank via intermediate states (if required) to the finished part is presented in Appendix 10.2. As WP model the initial state of the WP, the state prior to the machining of the actual setup, is used. As mentioned before, the first step of the IWS preparation is the definition of the TSV obstacle. For the presented verification, the TSV are created and added to the WP model manually. An automatic TSV and assembly model creation approach is presented in Section 6.1. The automatically generated TSV models, however, have a very detailed geometry, which increases the file size and computation time for the grammar application and reduces the clarity of the model pictures used to present the results.

For the verification, the complete rule set is loaded in the SGI. All application settings, including the rule application sequence, LHS matching and the setting of the parameter values, are set to random mode. This enables the fully automatic application of the grammar once initiated by the user. The collision detection is activated to not allow for collisions between the generated jaw designs and the WP itself. Collisions with the TSV are not allowed per se as it is defined an obstacle. Collisions or overlaps of non-obstacle solids, however, are generally allowed by the SGI if the collision detection is not activated. As will be shown later, such collisions can lead to invalid designs and are, hence, not allowed. The number of solutions, i.e. the number of design candidates to be generated, is set to ten. This number can be set higher

and is not limited by the SGI. However, ten solutions are sufficient to basically explore the design variations that the grammar can generate. When used as part of the automatic FD system a higher number of synthesized designs may be advantageous to yield a broader range of design candidates that can then be evaluated, as explained in Section 6.2.

In the following, two case-study parts, each in two different setups, are presented to show the results of applying the spatial grammar. These four examples show the range of basic jaw set shapes that the grammar can generate. For every example, two out of the ten generated solutions are presented. The solutions are selected manually to highlight the possible design variations and allow for a discussion of the capabilities of the grammar in the following section. Further data on the case-study, such as the average and maximum number of rule applications needed, is listed in Appendix 10.7.

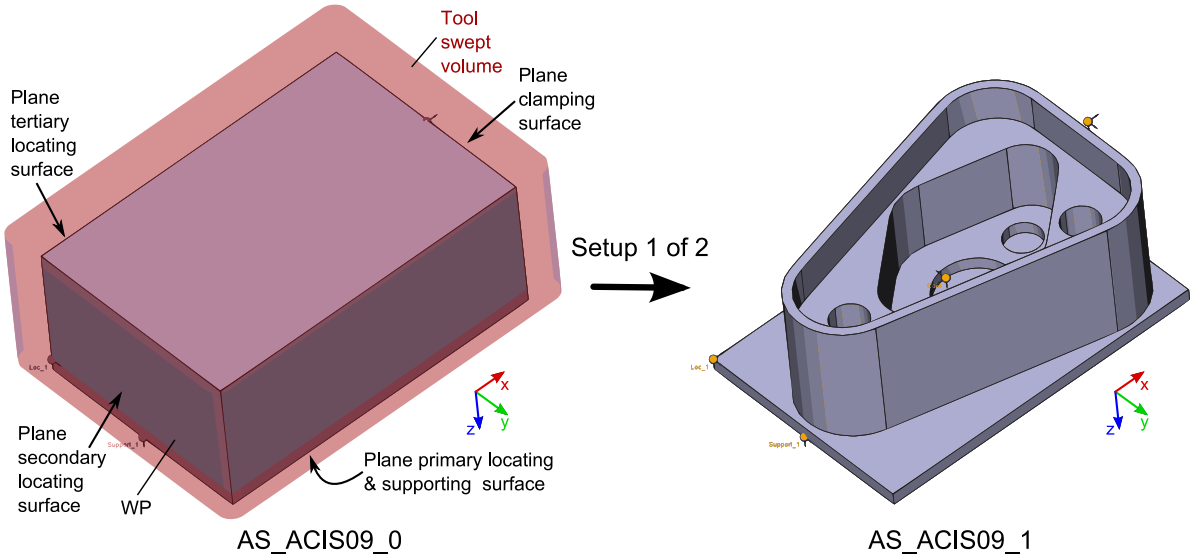



**Figure 5-13 Setup 1 of 2 of workpiece AS_ACIS09; left side shows blank with tool swept volume obstacle and all plane contact surfaces; right side shows resulting workpiece after machining of the setup**

Figure 5-13 shows the first setup of the WP AS_ACIS09 in which a cuboidal blank is transformed into an intermediate state. On the left side of the figure the prepared IWS is shown with the oriented blank WP and the TSV (in semi-transparent red)[4]. For better understandability of the setup all contact surfaces of the WP are highlighted. On the right side of the figure the intermediate WP model is shown that results from the machining of the setup and serves as the WP for the subsequent setup. The cuboidal blank has only plane and perpendicular contact surfaces and the primary locating and the supporting surfaces are the same. To fix this WP in the given setup, a jaw set consisting of a positive step-rest locating jaw and a step clamping jaw is required and should be generated by the grammar.

[4] The labels are included in the models but not further highlighted in the figures for better visibility of the shapes.

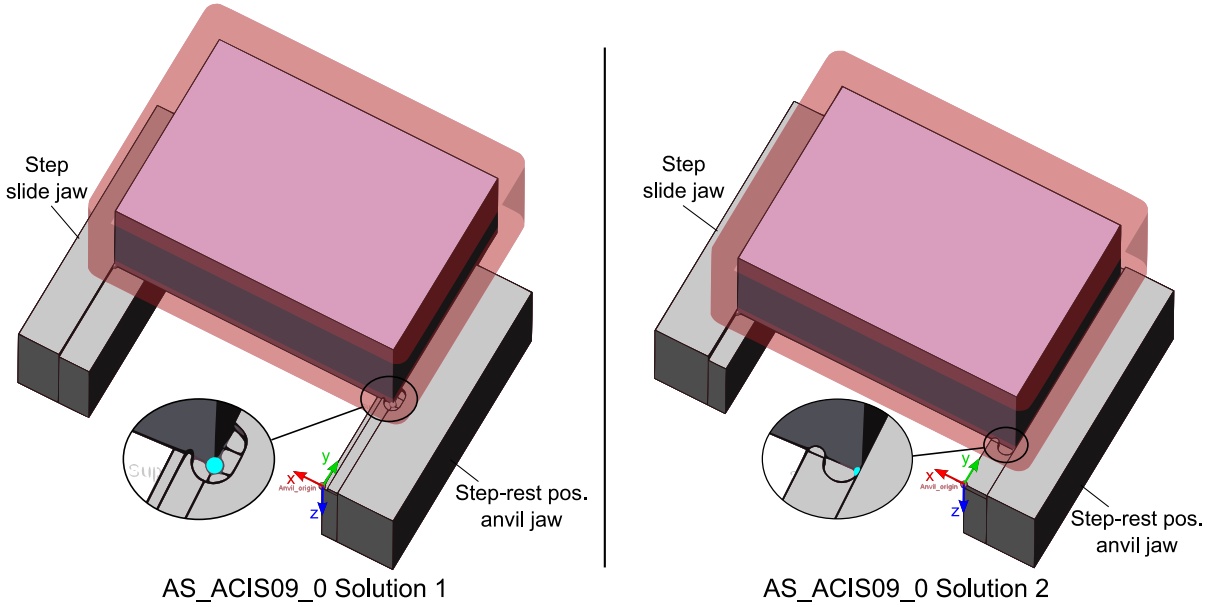



**Figure 5-14 Positive step-rest- & step-shaped jaw set candidates generated by the grammar for setup AS_ACIS09_0**

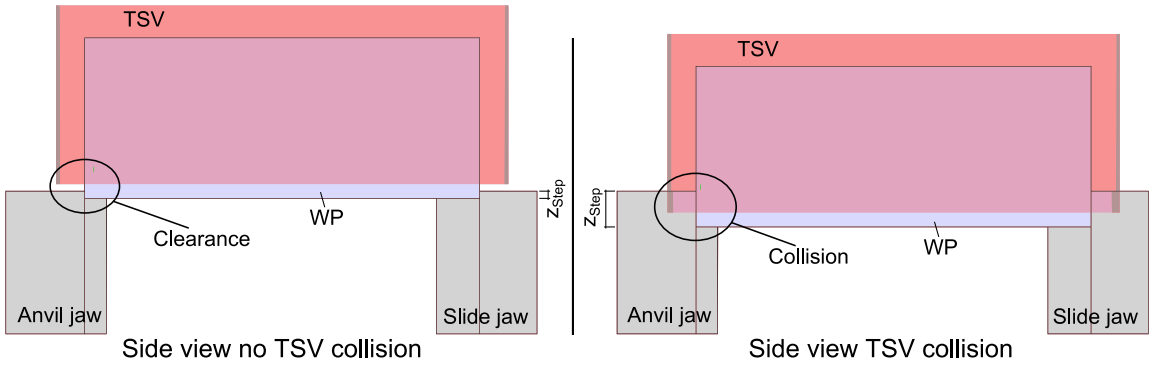



**Figure 5-15 Design space restriction caused by tool swept volume obstacle**

In Figure 5-14 two solutions for the previously presented IWS generated by applying the grammar in random mode are presented. As required by the IWS jaw sets consisting of a positive step-rest-shaped anvil jaw and a step-shaped slide jaw are generated. The resulting jaws vary in the geometric parameters of the step and the rest box. Further, the shape and dimension of the cutout vary depending on the previous parameters, as shown in the close-ups in Figure 5-14. When inspecting the two solutions it can be seen that the step depth only varies slightly. This is the case for all ten generated solutions and is induced by the design space restriction caused by the TSV obstacle. Figure 5-15 illustrates the limited feasible step depth

caused by the TSV. On the left of the figure a side view of the first solution is shown that does not collide with the TSV. On the right of the figure a manually made up jaw set with a higher step depth is used that causes a collision between the jaw and the TSV obstacle. Such a solution is not feasible and will not be created by the grammar.

Next, the jaw set design grammar is applied to the second and final setup of the demo WP AS_ACIS09. The IWS for this setup is presented in Figure 5-16. In this setup the intermediate WP (AS_ACIS09_1) is flipped upside down and the "baseplate" that remained from machining the first setup is removed by face milling. The cylindrical surface is used for locating while the opposite plane surface is used for clamping. These contact surfaces require a combination of a V jaw for locating and a step jaw for clamping as jaw set. Due to its location and the remaining height of the WP, the tool swept volume imposes no constraint on the design space.

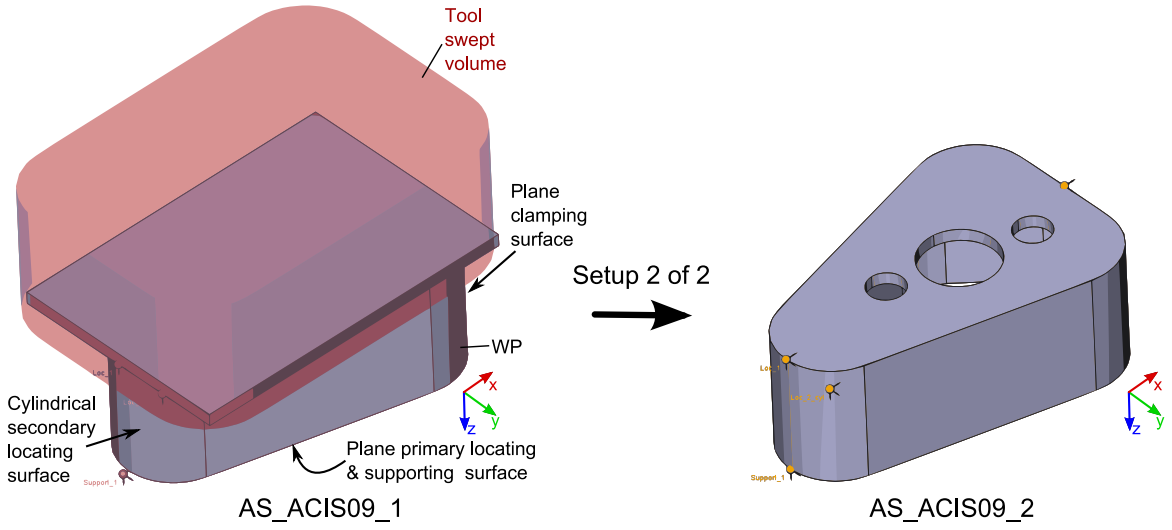



**Figure 5-16 Setup 2 of 2 of workpiece AS_ACIS09 with cylindrical locating and plane clamping surface**

Two solutions for this IWS are shown in Figure 5-17. The WP is shown as a wireframe model and the TSV, as well as the majority of the generated labels, are not shown in the figure for better visibility of the jaws. All generated jaw sets have a V-shaped anvil jaw, used to locate the cylindrical locating surface of the WP, and a step-shaped slide jaw, used to clamp the plane WP clamping surface. As the design space is less restricted by a TSV all possible geometric parameters, such as the opening angle of the V, the length and depth of the step or the wedges or the center-position of the V-cutout, vary among the solutions.

On the left side of Figure 5-18 a complete model of the second solution, including WP and TSV, is shown, as generated by the grammar. The middle of Figure 5-18 shows a cut bottom view of solution three that allows for the inspection of the contacts between the WP and the jaws. It can be seen that the rules set the y-placement of the wedges and the gap width such that the cylindrical WP surface is in contact with the sloped wedge surfaces, as required for accurate V-block locating.

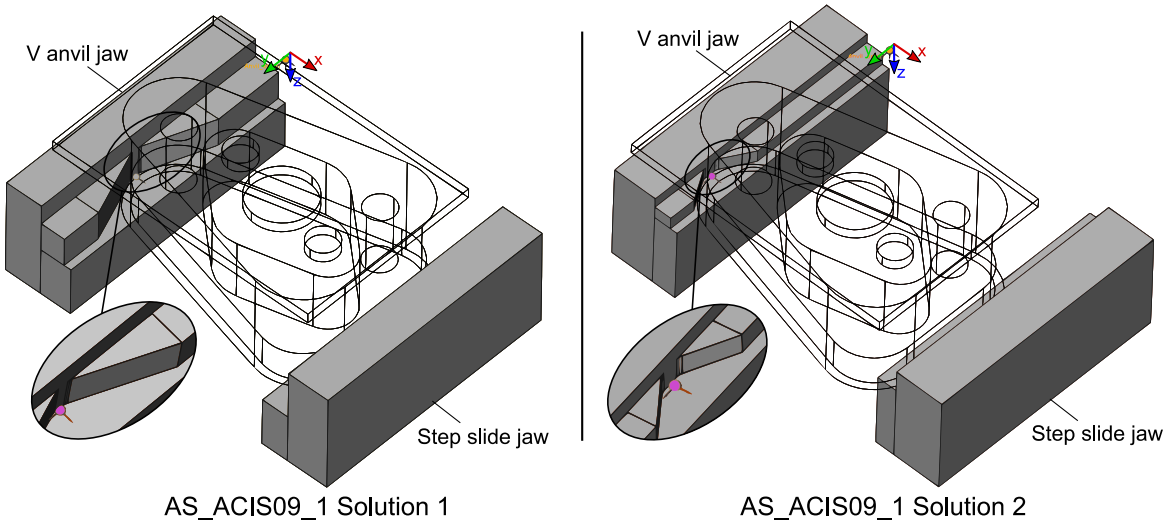



**Figure 5-17 V- & step-shaped jaw set candidates generated by the grammar for setup AS_ACIS09_1**

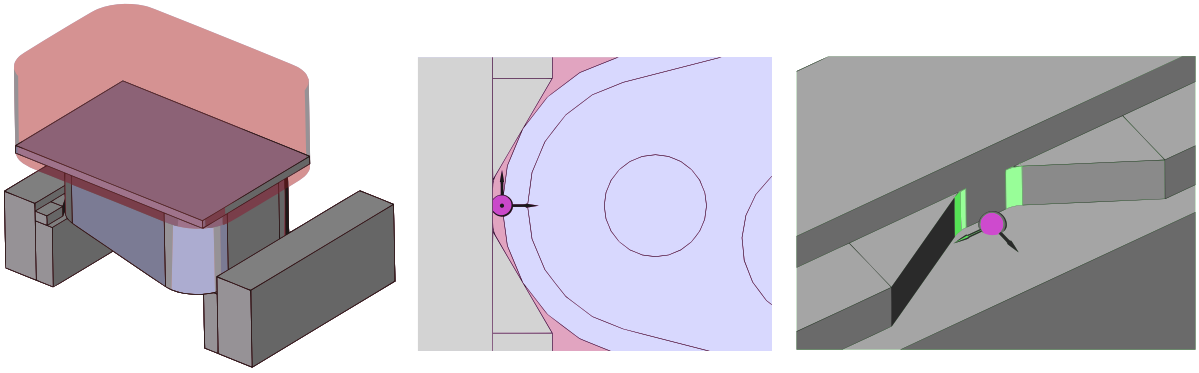



**Figure 5-18 Details of solution three for setup AS_ACIS09_1: Complete model generated (left), bottom view of the V-cutout (middle) and close-up of the generated fillets on the tips of the wedges (right)**

Further, the x-placement is set such that a minimum distance between the backing box of the wedges and the WP surface is maintained at all times to ensure a well-determined locating at two line contacts only. The right side of Figure 5-18 shows the generated fillets at the tips of the wedges highlighted. These fillets are generated using rule R03-1 and are accurately placed such that the rounding is tangent to the backing box and to the sloped surfaces of the wedges. The fillets ensure the machinability of the V-cutout with the available cutting tools using 2.5D machining.

The WP “Cap_Arm”, a cap of a robot arm, is used as second example to evaluate the jaw set design grammar. The part is again machined from a cuboidal blank in two setups. The first setup requires a step-rest & step jaw set again. As opposed to the first setup of the “AS_ACIS09” part, the secondary locating surface of this part is in positive y-direction, making a negative step-rest jaw necessary for locating. Figure 5-19 shows the IWS for the first setup of the part with highlighted contact surfaces and the TSV on the left and the resulting intermediate WP state on the right. The WP has all plane contact surfaces and the primary and secondary supporting surface are the same.

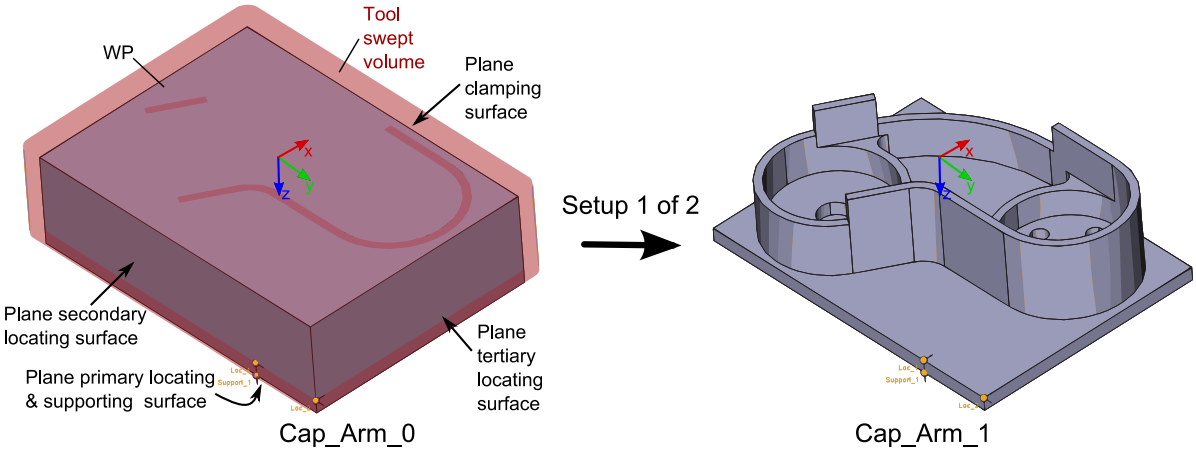



**Figure 5-19 Setup 1 of 2 of workpiece Cap_Arm with all plane contact surfaces and secondary locating surface in positive y-direction**

Two solutions for this setup are shown in Figure 5-20, with the WP depicted as a wireframe model and the TSV depicted as semi-transparent red solids. The figure illustrates that the grammar is able to generate jaw set designs with different parametric variations and cutouts to ensure machinability. The rest box is built on the left side of the anvil jaw, as required. The width of the resting boxes is small and does not vary a lot in all generated examples to keep the center of mass of the WP placed within the jaw width. This prevents the WP from tilting out of the jaw set when the vise is not engaged. Further, the outside face milling operation causes a restriction of the feasible depth that the step-cutouts of the jaws can have. Thus, the step depth varies only slightly. The other design variations are equal to those presented for setup AS_ACIS09_0 and are, therefore, not further discussed here.

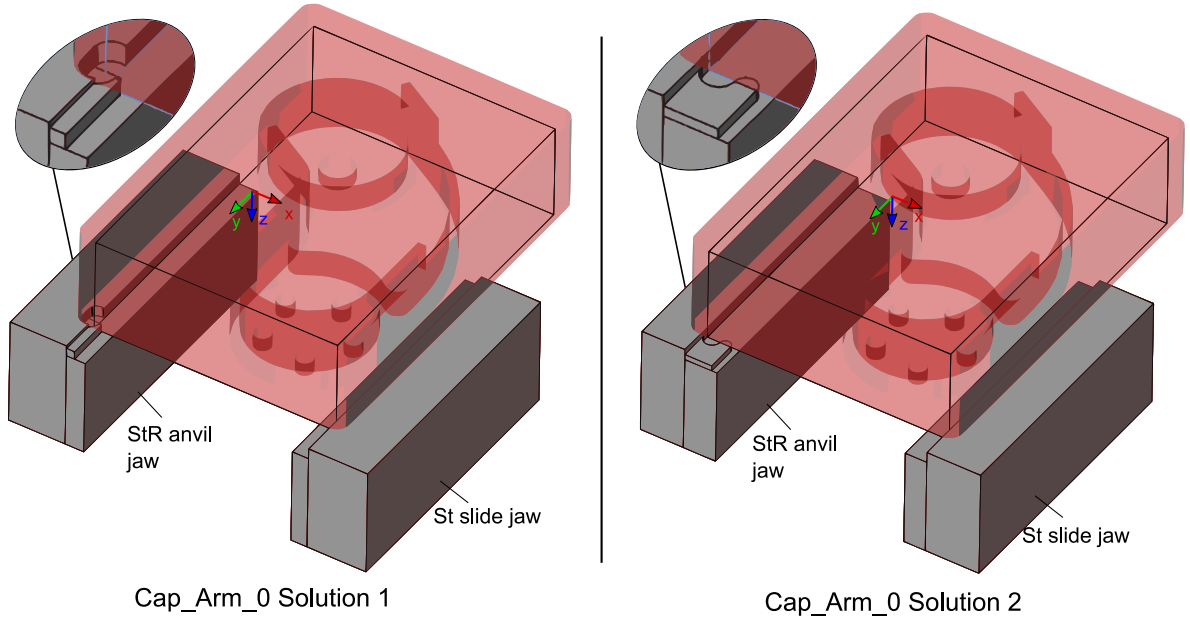



**Figure 5-20 Step-rest- & step-shaped jaw set candidates with secondary locating surface in positive y-direction generated by the grammar for setup Cap_Arm_0**

The second and final setup for the WP Cap_Arm, Cap_Arm_1, imposes the last possible combination of contact surface shapes: a cylindrical secondary locating surface with a cylindrical clamping surface. The IWS for this setup is shown in Figure 5-21. This setup also has two different surfaces as primary locating and supporting entities. Special attention has to be paid when orienting the WP and TSV in the IWS preparation phase. As stated in the setup rules, the center points of the cylindrical contact surfaces have to align on one line parallel to the x-axis in order for the grammar to generate feasible results.

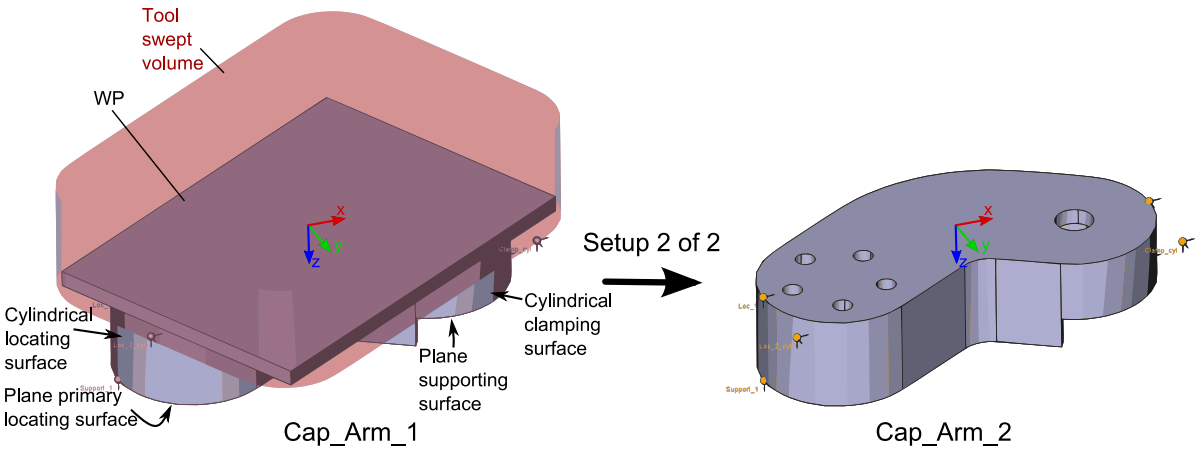



**Figure 5-21 Setup 2 of 2 of workpiece Cap_Arm with cylindrical locating and cylindrical clamping surface**

As presented before, a setup of this kind in combination with a random rule application sequence allows for two different jaw set shapes: a combination of V & V or V & step jaws. Among the 10 generated solutions are six V & step sets and four V & V sets. Figure 5-22 shows two selected solutions, one with a V & step combination and one with a V & V combination. The WP is shown as a wireframe model and the TSV is not depicted for better visibility.

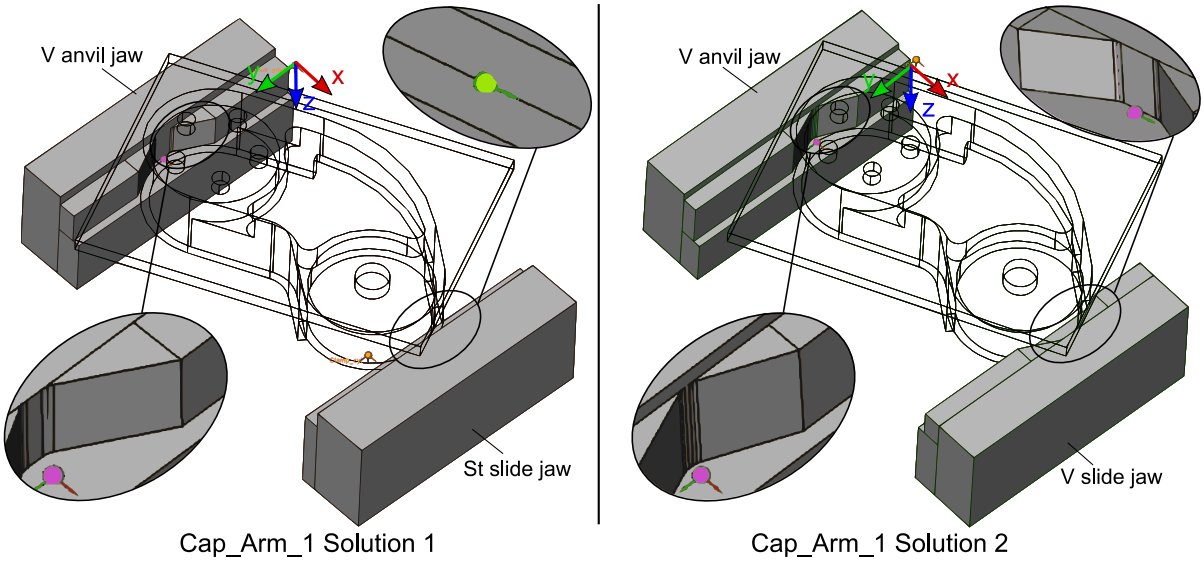



**Figure 5-22 V- & step-shaped and V- & V-shaped jaw set candidates generated by the grammar for setup Cap_Arm_1**

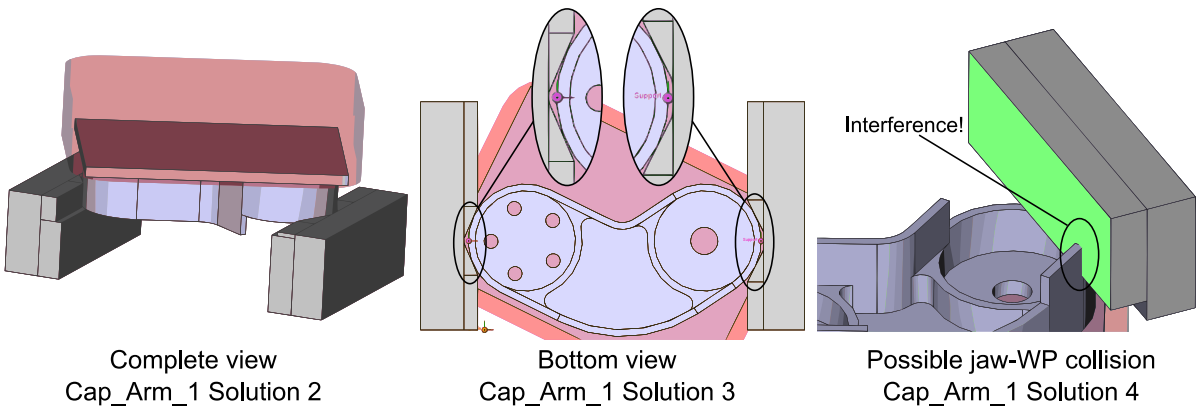



**Figure 5-23 Details of solutions for setup Cap_Arm_1: Complete model (left), bottom view of the V-cutouts (middle) and close-up of a possible jaw-workpiece collision (right)**

The results show that both the V & step and the V & V jaw sets are generated correctly. The different height of the supporting surfaces, which leads to an interdependency between the

minimum and maximum range values of the step depth of the anvil and slide jaw, is handled correctly. The different widths of the wedges in case of V & V designs, which cause an interdependency of the maximum y-placement of the wedges on both jaws, are also processed correctly. In Figure 5-23 a few details of the solutions, including the complete model of solution two as generated by the grammar (left side) and a bottom view of solution two, highlighting the accurately established contacts between the cylindrical WP surfaces and the sloped surfaces of the V-blocks (middle) are shown. The right side of Figure 5-23 shows a collision between the supporting box of the slide jaw and the WP. Such collisions can occur when other surfaces than the initially labeled WP contact surfaces are present in the possible design space of the jaws. These collisions would lead to infeasible jaw set designs. By activating the collision detection in the SGI rule application process this kind of collision is prevented. As this setting was activated for the generation of the results, the example shown in the figure is made up for illustration only.

Next, the ability of the grammar to deal with plane but angled tertiary WP locating surfaces is verified. This is tested using the second setup of the part “Pentagon” and the second setup of part “Angled_Socket”. In the setup Angled_Socket_1, the WP has a plane tertiary locating surface facing (partially) in the negative y-direction and meeting the secondary locating surface at an acute angle. Opposed to this, the plane tertiary locating surface of setup Pentagon_1 faces (partially) in positive y-direction and meets the secondary locating surface at an obtuse angle.

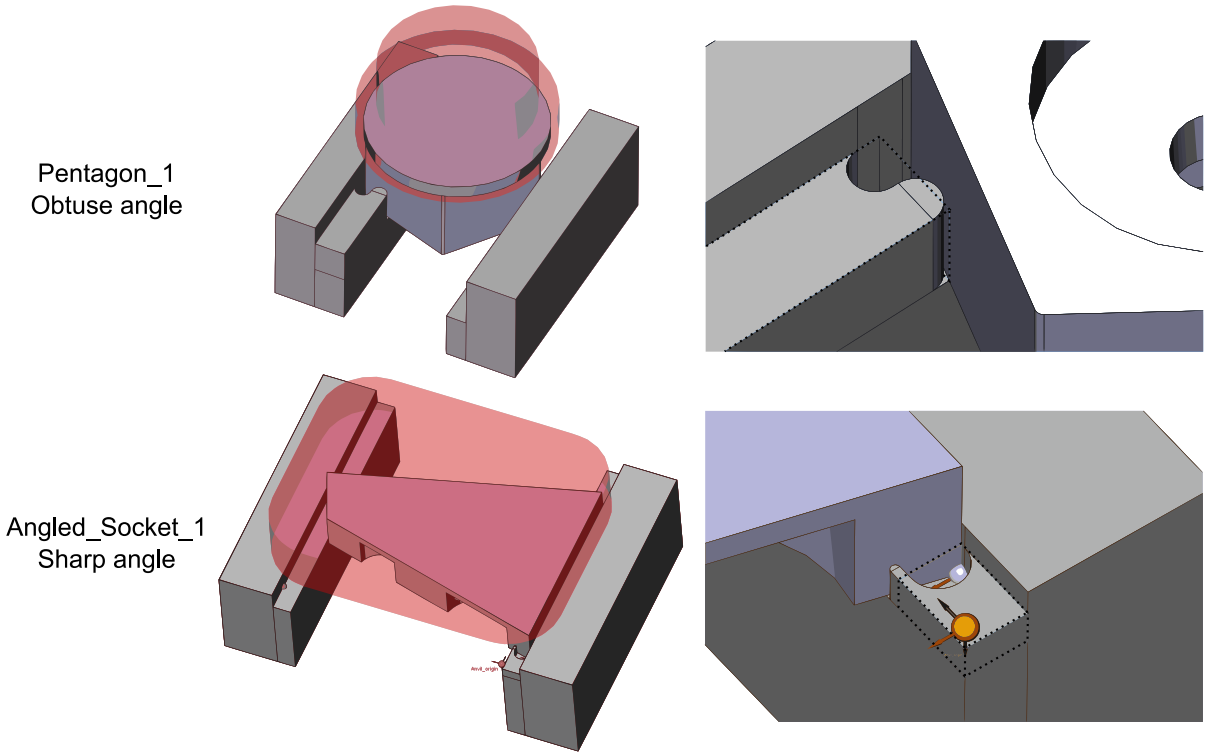
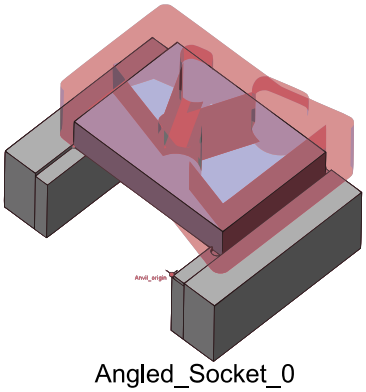
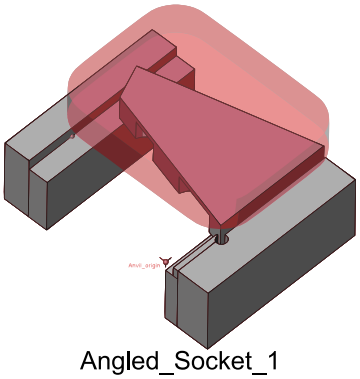


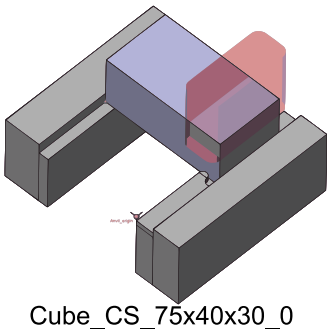

**Figure 5-24 Shortening or extension of tertiary locating element of step-rest jaw for workpiece with obtuse or acute angle between secondary and tertiary locating surfaces**

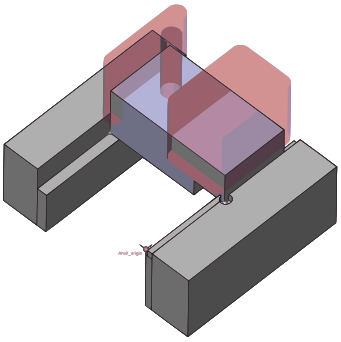
The top of Figure 5-24 shows a solution for the setup Pentagon_1 in full and a close-up of the tertiary locating contact. The contact element of the step-rest jaw is not only cutout and rounded but also shortened in comparison to the original locating element (rest box) that was built in rule R00-X. For better visibility this box is shown as wireframe with black dotted lines. It can be seen that the half-cylinder that builds the rounding is placed exactly to be in contact with the tertiary locating surface of the WP surface. The bottom of Figure 5-24 shows a solution for setup Angled_Socket_1. In this setup the contact element of the jaw needs to be extended to bridge the gap to the angled tertiary locating surface of the WP. The original rest-box built in rule R00-X is again depicted as black dotted wireframe. The two examples show that the grammar can handle both obtuse and acute angles and step-rest jaws with positive and negative y-locating direction.

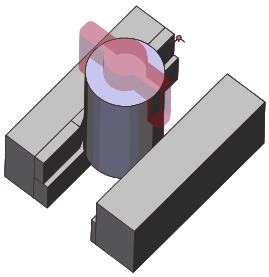
The grammar was successfully applied to all other case-study parts (from the list of demo parts) for further evaluation. This includes setups with a positive z-distance between the primary locating and the supporting surface, setups with obtuse angles and positive y-locating direction as well as with acute angles and negative y-locating direction, which were not yet presented. Detailed results of these case-studies are not presented here due to the restricted space. However, Figure 5-25 shows an overview of one solution for each WP and setup that has not yet been covered. Additional data, like the average or maximum number of rule applications for all setups, is given in Appendix 10.7.

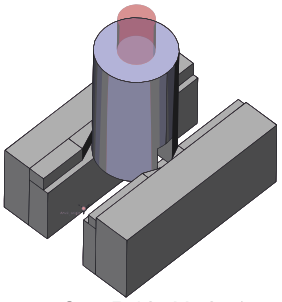

Angled_Socket_0

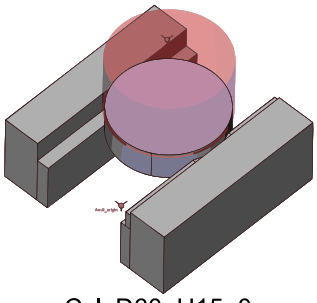

Angled_Socket_1

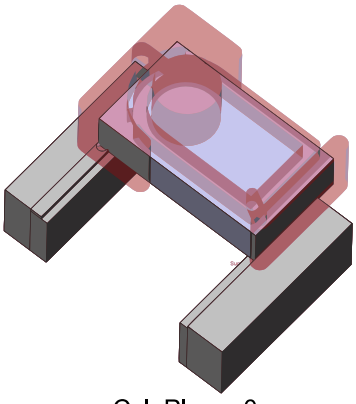

Cube_CS_75x40x30_0

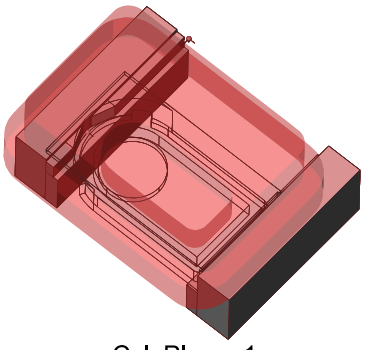

Cube_CS_75x40x30_1

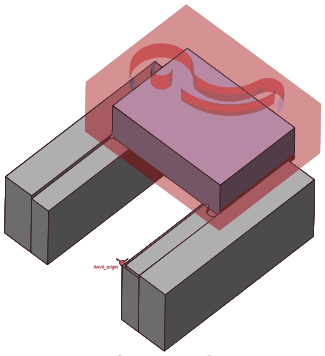

Cyl_D40_H70_0

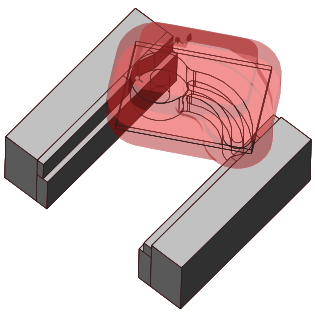

Cyl_D40_H70_1

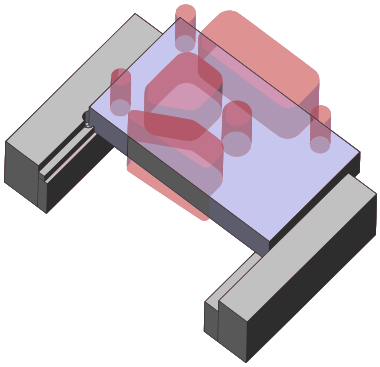

Cyl_D60_H15_0


Cyl_Plane_0


Cyl_Plane_1


Half_Butterfly_0


Half_Butterfly_1


Holder_0

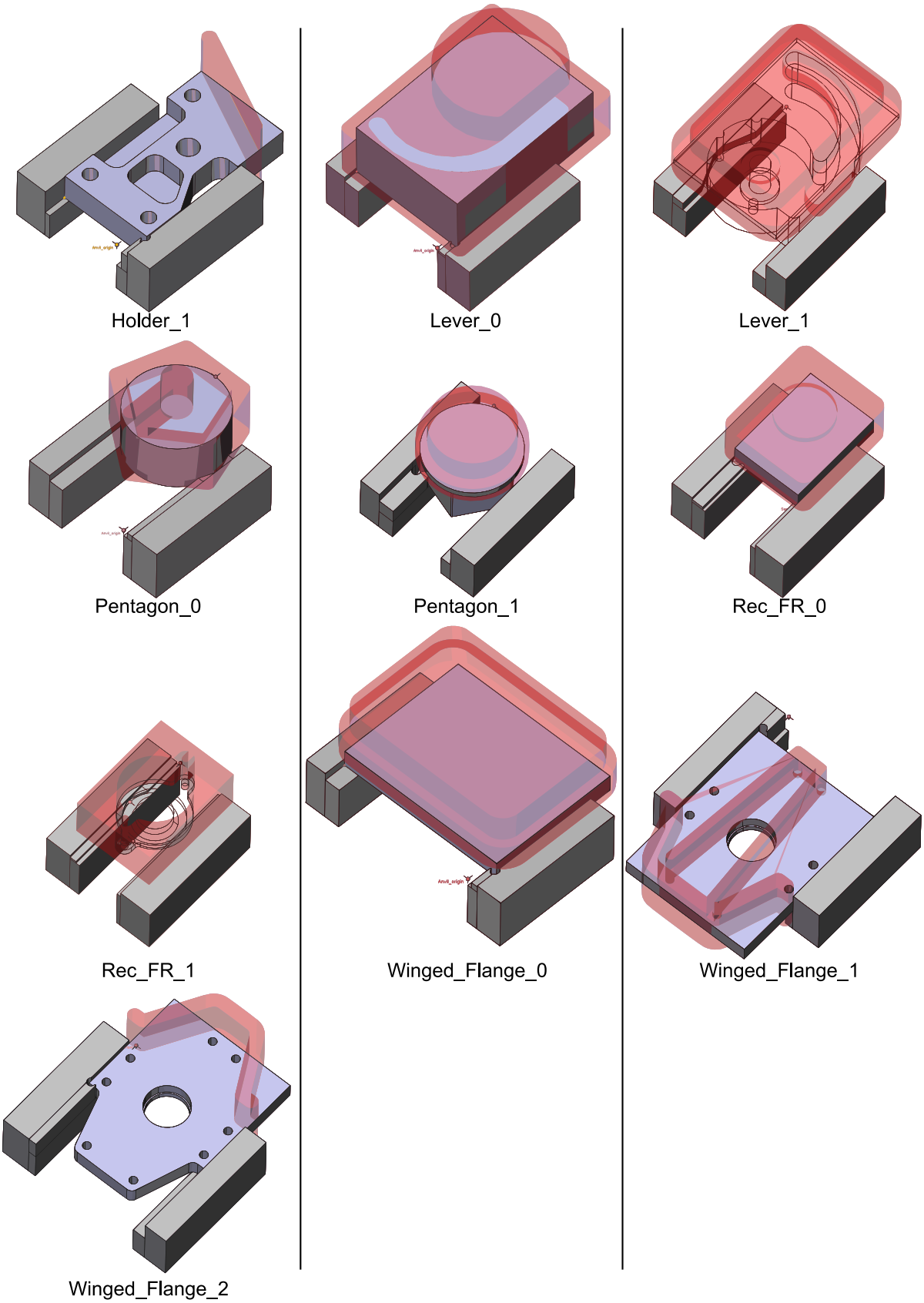



**Figure 5-25 One jaw set candidate generated by the grammar for each case-study workpiece (continued)**

## 5.5 Discussion

The case-studies show that the developed spatial grammar is able to generate jaw set designs for the complete WP spectrum considered in this research. All generated jaw designs meet the outer dimensions of the blank jaws and respect the minimum and maximum machinable volume, as presented in Figure 5-5. Driven by the set of labels in the IWS, the correct locating principle and basic jaw set design is chosen automatically when the grammar is applied. Despite the variations in shape and parameters, the WPs are always accurately positioned in contact with the locating, supporting and clamping surfaces of the jaws. Design constraints, such as the correct placement of the center of mass of the WP within the y-boundaries of the jaws to ensure stable loading/unloading, are satisfied in all solutions. Further, the geometry and placement of the cutouts and the fillets, as shown in the close-ups of the solutions, are established correctly to allow for the machining of the jaw set designs with the given tools and processes. This is similarly observed for the interdependency of the step lengths or the supporting surface heights of the locating and clamping jaw of each synthesized set. Some examples, such as the setup Cap_Arm_1, also show that the ranges of the design variables set by the constraints are fully explored. Among the ten generated solutions are designs with the minimum as well as the maximum allowed wedge length or V-block opening angle. The case-studies prove that the rule set works correctly and that the parametric dependencies and ranges for the design parameters, as defined by the requirements, are set right.

The rule set was developed in a bottom-up approach with several iterative cycles. This is due to the complicated nature of the grammar caused by the embedding of the design requirements and the lack of a detailed, systematic (spatial) grammar development methodology. Explicitly transferring the requirements into geometric constraints on the available design parameters ahead of the development of the formal grammar could reduce the need for iterations and speed up the development process.

The approach of using a set of labels in the IWS to provide the necessary data about the placement and shape of the WP contact surfaces, instead of using the WP solid model itself, allows the grammar to generate jaw set designs for non-predefined WP shapes while keeping the rule set small, with a total of 12 rules. Other options to deal with the non-predefined WP shapes, such as a dissection of the WP into primitives using a spatial grammar or directly addressing the faces of the WP models as 2D entities, were explored without any success. The approach of using labels to provide all necessary WP data makes the presence of the WP model itself in the IWS unnecessary. However, leaving the WP model in the IWS has some advantages: the grammar can generate assembly models of the jaw set and WP that can directly be used as input for the subsequent design analysis, as shown in Section 6.2. Additionally, the WP model can be used to detect collisions with the jaws during their generation process.

The grammar is modeled such that the possible jaw set shapes are restricted to only four versions. With parametric variations of these jaw set shapes all case-study WPs can be located and clamped accurately for all necessary setups, given that the setups are planned in adherence to the presented guidelines. The shapes of the WP surfaces that are in contact with the jaws define which jaw set shapes are applicable. For WPs with either a planar secondary locating surface or a cylindrical secondary locating surface, in combination with a planar clamping surface, the selection of the jaw set design is deterministic, as only one design can be chosen.

For WPs with a cylindrical locating and clamping surface, the grammar can randomly choose between two jaw set shapes, making the selection non-deterministic.

Every jaw shape has several design parameters that are, within predefined ranges, randomly set by the SGI during the rule application. This way, design variants of the specific shape are generated and the solution space is explored. The solution space is restricted by the TSV, which is part of the IWS, the WP itself and the design constraints. Collisions between the generated jaws and the TSV or the WP are not allowed. This ensures that all generated design candidates are interference-free. As the results show, the generated jaw sets meet all design constraints. This holds for all jaw shapes and jaw set combinations and includes all fixed and also all flexible requirements given in Table 5-2. The requirements are, therefore, fully embedded in the grammar.

Although the generated design candidates meet the constraints, they are not guaranteed to be structurally feasible or equally well performing solutions. Considering the solutions of the setup AS_ACIS09_0, shown in Figure 5-14, it can be seen that they have different contact area sizes due to the y-placement of the WP within the jaw set. This does have an effect on the momentum caused by the machining forces and the resulting stress distribution in the jaw and the WP. Designs with bigger contact areas will likely perform better with regard to WP stability or WP deformation. This is similarly seen in the results for setup Cap_Arm_1 presented in Figure 5-22. From a locating perspective, all generated results for this example are feasible. The V & Step jaw set designs lead to well-determined locating and the V & V designs to over-determined locating. Although well-determined locating is to be preferred according to fixture design guidelines, the V & V designs are likely to outperform the V & Step designs with regard to withstanding momentum around the z-axis caused by the machining forces. However, it is not the aim of the grammar to verify the design candidates or evaluate their performance but only to generate them. The verification has to be done in the subsequent FE analysis, as presented in Section 6.2.

By restricting the possible designs of the jaws to a set of basic shapes one of the main benefits of spatial grammars, the possible creation of unexpected solutions by generating different topologies, is not taken advantage of. However, an uncontrolled or less-controlled design generation is not possible if the design requirements and constraints are to be satisfied by the synthesized designs. If the constraints are loosened during the synthesis phase, more designs will emerge that do not meet the requirements and, therefore, fail their purpose. Determining designs that do not meet the requirements, i.e. infeasible designs, in subsequent process steps requires design verification approaches that are often computationally intense and time consuming and only add to the complexity of the AFD system. Thus, including the requirements in the synthesis and restricting the possible solution space is considered advantageous for the given design task. Additionally, the possible parametric variations of each jaw set shape still allow for the generation of sufficient design variants to find one or more feasible solutions for all case-study parts. Therefore, no advantages arise from an unrestricted jaw shape generation in this case.

For the presented synthesis task other computational synthesis approaches could be used alternatively to a spatial set grammar, such as graph grammars or a combination of a rule-based system with CAD-based parametric modeling. However, none of these approaches is thought

to have any advantages over the use of a spatial set grammar. On the contrary, the generation of a combination of a rule-based system for the case differentiation plus parametric modeling is thought to be more laborious than the spatial grammar development. The developed grammar and used SGI can successfully and automatically solve the design synthesis task at hand with a reasonable development effort. Due to the open-source framework used, the grammar can easily be extended and reused or altered for other scenarios.

With the spatial grammar, the jaw models are created in a mainly additive process around the WP model. An alternative option considered was the use of a subtractive process in which the WP model is subtracted from a set of raw jaw models. However, as only a WP spectrum but not the exact shape of each WP model is known at the time of the rule development, this would require a workaround. One option is to split the WP model into primitives using another set of rules. The rule set of the jaw set design grammar can then scan for matches based on these primitives. Nevertheless, the dissection of complicated WP models into primitives is a non-trivial task and results in a vast rule set. Additionally, the subtraction of the WP models from the raw jaw models leads to fitted, over-determined contacts between the WP and the jaws, which would need to be resolved with more rules. Hence, the additive process was preferred.

The design generation process happens fully automatically once initiated by the user in the SGI. This includes defining the values of free parameters within the given ranges, LHS matching and defining a rule application sequence. The use of fully state controlled rules enables automatic termination of the application process once no more rules are applicable. However, the tasks of opening the IWS in the SGI, selecting the set of rules to apply, setting the number of rule applications and solutions to be generated as well as the target path to store the solutions at and finally starting the rule application process have to be carried out manually so far. Automation of this process via a *Python* script can be realized in *FreeCAD* and *Spapper*, but has not been pursued in this work. As illustrated in the AFD system process shown in Figure 3-2, the generated FD candidates are only stored temporarily a first. If a synthesized candidate is selected as final design in the verification phase, the created jaw models and FD data are added to the repository and the AFD ontology. This also allows for the reuse of a single jaw, as opposed to the jaw set, to build candidates by ontological reasoning for future FD tasks.

One issue that arose during the case-study is that in case a WP with cylindrical locating and clamping surfaces and a TSV that tightly restricts the design space is processed, the grammar is more likely to produce V & Step jaw set designs instead of the equally possible V & V designs. This can be seen in the results of setup Pentagon_1 or setup Cyl_D60_H15_0 in Appendix 10.7. Out of the ten generated solutions only one was a V & V design. The reason is that V & V jaw sets consist of two more solid primitives that are in contact with the WP, the two wedges at the slide jaw, than V & Step jaw sets. Therefore, collisions with the TSV are more likely to occur when applying rule R00-4 (V & V) than when applying R00-3 (V & Step).

## 5.6 Conclusion

The development and evaluation of a spatial grammar for the synthesis of jaw set designs for given WP models and setup plans was presented in this section. The grammar is able to generate parametric variations of different jaw set shapes that can be used to accurately locate and clamp any of the WPs in the given geometric WP spectrum.

The grammar, in combination with the SGI used, allows for the semi- and fully-automatic generation of new fixture configuration designs. This is necessary for the AFD system in case no feasible designs can be found among the previously known or existing ones by ontological reasoning. A number of design requirements are embedded in the grammar. This way, it is ensured that the synthesized designs satisfy the requirements, which reduces the need for subsequent computational analyses in the design verification phase. An important aspect is the use of the TSV model as a geometric obstacle in the synthesis process. It allows for the generation of jaw set designs that are guaranteed to be free from tool-fixture-interferences. Unlike the fixture configuration designs generated through ontological reasoning, the synthesized candidates, therefore, do not require any further interference analysis. However, all design candidates, the ones generated by the spatial grammar and the ones determined by ontological reasoning, require structural analysis for evaluation. The necessary analyses for fixture design verification and evaluation are presented in the next section.

# 6. Fixture Design Verification

To ensure the safe operation of the fixture design (FD) candidates found by ontological reasoning or created by the spatial grammar they need to be verified. The presented approach focusses on two aspects: 1) that the fixture designs are free from interferences between the cutting tool and the fixture and 2) that among the available candidates the one causing the least workpiece (WP) deformation is used. The verification is carried out using an interference analysis and the evaluation using a finite element analysis (FEA). Special emphasis is placed on the ability to automate both analyses.

The interference analysis checks for collisions between the cutting tools and the reconfigurable fixture components. Such interferences need to be avoided to not alter or destroy the fixture. The analysis is based on the tool swept volume (TSV) approach, where the solid volume that the cutting tools require in 3D space for carrying out the machining operations is built by sweeping operations and used to determine the interferences. The use of this approach in the given system has the huge benefit that the built TSV models can be used as design space restricting obstacles in the spatial grammar-based design synthesis. This is discussed in Section 5.3.2. All FD candidates generated by the spatial grammar for the specific fixture problem are, therefore, guaranteed to be collision free. Thus, the interference analysis is only required for the candidates determined by ontological reasoning but not for those generated by the spatial grammar.

All FD candidates that are found to cause an interference are not further processed or used in the FD process. All interference-free FD candidates are further analyzed for WP deformation. As WP deformation correlates directly with the part accuracy the FD that causes the minimum deformation is to be preferred. Thus, the deformation analysis is used to evaluate the candidates. The analysis is based on the finite element method and is carried out for the candidates generated by the spatial grammar and for the ones generated by ontological reasoning that passed the interference analysis. The analysis uses the assembly models generated by the spatial grammar or as part of the interference analysis as geometric input. Figure 6-1 illustrates the link between the candidate generation and analysis processes.

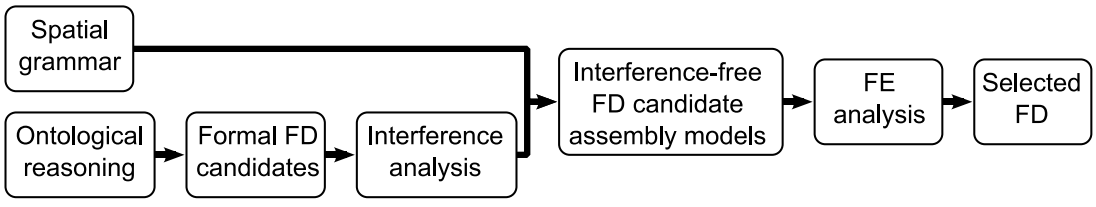



**Figure 6-1 Link between different possibilities of fixture configuration design (FD candidate generation) and analyses for FD verification**

## 6.1 Interference Analysis

The interference analysis is necessary to verify that the generated fixture candidates do not collide with the cutting tools during the machining process. The approach is based on detecting intersections of the 3D models of the fixture components and the TSV model. The TSV represents the volume(s) that the cutting tools require to carry out the travelling and cutting operations that are necessary for the machining of the workpiece. The TSV is generated based on the tool used and its travelling paths. This information is extracted from the NC file of the setup. In parallel to the TSV generation, the assembly models of the candidates are created. The assembly models consist of the WP and the fixture components determined by the ontological reasoning. The TSV in combination with the assembly model of each FD candidate are then used to detect interferences between the TSV and the fixture components. Only collision-free designs are passed on to the structural analysis. Keeping the idea of an automated fixture design (AFD) system in mind, the analysis needs to work with minimal user (or agent) interaction and the input and output data needs to be stored or linked in the AFD ontology. The basic algorithm for the implementation of the approach is shown in Figure 6-2. The four main process steps shown in the dark grey boxes are explained in detail in the following. Note that the four steps are presented here in one section as they are necessary for the interference analysis. However, considering the overall AFD process presented in Figure 3-2, the tasks happen at different points in time. The G-code interpretation and the TSV generation must happen at an early stage, as the TSV are required as obstacles for the FD synthesis presented in Section 5. The assembly model generation on the other hand is done after the fixture configuration design based on ontological reasoning and before the actual interference test.

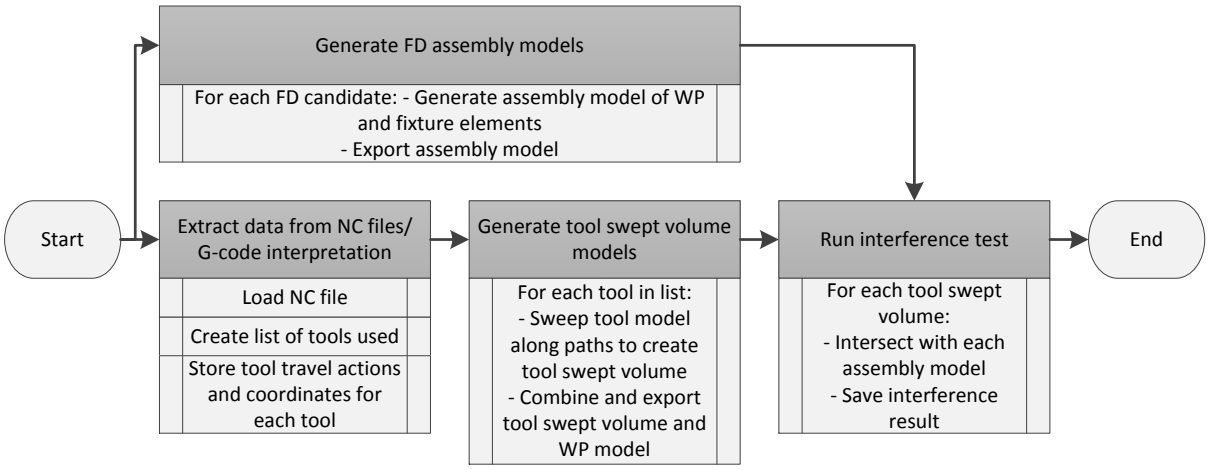



**Figure 6-2 Basic algorithm of interference analysis**

### 6.1.1 Automated Assembly Model Generation

The ontological reasoning determines possible FD candidates for new workholding jobs built from existing fixture components. The FD candidate representation includes the file-paths of the 3D models of the WP and the vise jaws used. To enable an interference analysis, assembly

models of these FD candidates need to be generated automatically. The process is independent from but necessary for the interference analysis. The resulting assembly models are also necessary for the deformation analysis, as presented in Section 6.2. In these assemblies, the WP and the fixture components need to be correctly aligned. Not the whole fixture but only the exchangeable components, i.e. the anvil and slide jaws, are used in the assemblies. This is done to keep the file size to a minimum and does not have a falsifying effect on the interference analysis, as interferences in the 2.5D machining scenario are most likely to happen between the cutting tools and the fixture components that are in contact with the WP. The individual models of the fixture components and the WP are called using the file path information stored in the AFD ontology.

**Table 6-1 3D labels added to the jaw models with the surface the label relates to and the translational parameters that need to be set to store the geometric information of the jaw surface (in the vise COS)**

| Label name | Jaw shape | Set parameters - information stored |
|---|---|---|
| Anvil_origin | All anvil | TransX/Y/Z – Top, front, right corner of anvil jaw (when frontally facing the jaw) and defined origin of machine COS |
| Cont_Sup_1 | All anvil | TransZ – z-position of supporting surface of anvil jaw |
| Cont_Loc_1 | StR anvil | TransX – x-position of primary locating surface |
| Cont_Loc_2 | StR anvil | TransX – x-position of front of rest box<br>TransY – y-position of secondary locating surface<br>TransZ – z-position of top of rest box |
| V_center | V anvil/<br>V slide | TransX – x-position of tip of wedges<br>TransY – y-position of center of V-cutout |
| V_wedge_tip | V anvil/<br>V slide | TransY – y-position of tip of wedge (closer to the origin)<br>Yaw – differentiation of V-shaped anvil (yaw=0°) and slide (yaw=180°) jaws |
| V_wedge_out | V anvil/<br>V slide | TransX – x-position of front of V-cutout<br>TransY – y-position of shoulder of wedge (closer to the origin)<br>TransZ – z-position of top of wedge |
| Cont_Clamp | St/ V slide | TransX – x-position of clamping surface of slide jaw<br>TransY – y-position of left side of slide jaw (when frontally facing the jaw) |
| Cont_Sup_2 | St/ V slide | TransZ – z-position of supporting surface of slide jaw |

In the assembly model the WP serves as fixed element, meaning that the anvil and the slide jaw are positioned with respect to the WP. This procedure is necessary as the NC files are generated with reference to the WP COS and position. Thus, the resulting TSV are also generated with

respect to the WP. To be able to add the TSV to the assembly model in a subsequent process step without the need for any re-positioning, the WP also needs to be the reference for the assembly models.

As the used geometric kernel does not currently provide an open-source 3D constraint solver, another approach is required to correctly place the jaws in contact to the WP. As shown in the AFD process scheme (Figure 3-2) and explained in Section 5.3.3, the contact surfaces of each WP are labeled according to the setup plan using the *Spapper* shape grammar interpreter (HOISL 2012). Additionally, all jaw models that are in the model repository are annotated with labels. The labels are used to mark the contact surfaces of the WP and jaws and are generated by the spatial grammar as part of the computational synthesis of new jaw set designs. The spatial relation between the labels of the WP and the labels of the jaws are used to compute the resulting jaw positions. The labels of the WP and the information they store are listed in Table 5-3. The labels of the jaws are listed in Table 6-1 and depend on the jaw shape and whether the jaw is used as anvil or as slide jaw. Figure 6-3 shows an example of each jaw type with the labels highlighted.

The label data is explicitly stored as part of the WP and jaw models. To compute the placement of each jaw, the data of all labels of the assembly parts is extracted from the files. Most placement values can be directly set. The x-value of Loc_1 defines the x-value of Cont_Loc_1, and so on. For V-shaped jaws, the positioning has to take the contact establishment at the sloped surfaces into account. This is done using trigonometric functions as described in Figure 10-1. This way, a placement vector is calculated for each jaw. Then a new assembly model is built by importing the WP and jaw models and placing the jaws according to the calculated placement vectors. The assembly model of each FD candidate is then saved and the name and location of the generated assembly model is added to the FD candidate data.

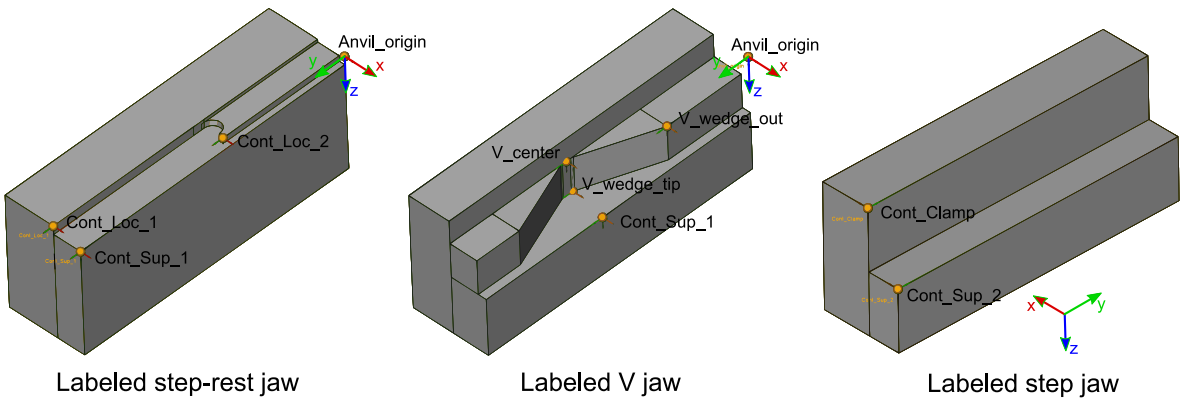



**Figure 6-3 Labeled step-rest anvil jaw (left), V-shaped anvil jaw (middle) and step slide jaw (right)**

## 6.1.2 Extraction of Data from NC Files

To create the TSV, the models of the cutting tools are swept along their respective travelling paths. Thus, the tools used and their travelling paths need to be known. This information is

extracted from the NC file of the setup. NC files can be created manually or (semi-) automatically[5] using computer-aided manufacturing applications. For the presented research, the NC files are created manually. A NC file is a text file that contains all commands and parameters necessary to drive a computer numerically controlled machine tool. The commands are given in so called G-code. An excerpt of an NC-file illustrating some lines of G-code is shown in Figure 6-4.

```
⋮
N340 T2 M06
N350 D2
N360 G0 G90 G40 G17
N370 G94 F318 S3183 M3
N380 G64 SOFT
N390 G1 X56 Y25 Z -8 F318
N400 Z2
N410 X50
N420 X43.604
N430 X30.883 Y63.162
N434 G2 X21.396 Y70 I-9.487 J-3.162
⋮
```

**Figure 6-4 Excerpt of NC file showing G-code**

G-code consists of different command classes, such as G-, T- or M-commands. M-commands represent different machine functions and T-commands represent tool selections/changes. G-commands represent different actions, such as movements, coordinate system (COS) definitions or triggering of certain machine functions. The NC file is interpreted line-by-line and divided into tokens. A token consists of a letter and a numerical value, e.g. “G2” or “X21.396”. The tokens are used to create a list of objects, one for each tool (T*x* token). Each object contains a list of actions carried out with that tool (G*x* tokens) and the variables for each action (X,Y,Z,… tokens). For the generation of the TSV models, only the T-commands and the G-commands that cause a movement of the cutting tool are required. Therefore, only the tokens for the following commands are used initially:

- T*x* – selection of a cutting tool
- G0/G00 – Rapid translation to a target position
- G1/G01 – Linear interpolation to a target position
- G2/G02 – Clockwise circular interpolation to a target position with the center-point of the circle given

[5] A method for the automatic generation of NC files developed in the CogMaSh project is presented by ERTELT (2012).

- G3/G03 – Counterclockwise circular interpolation to a target position with the center-point of the circle given

This list of G-commands covers the most common movement commands and all commands required for the machining of the case-study parts used in this work. Other tokens appearing in the NC files are not considered. Additionally to the command tokens, the following parameter tokens are used:

- X/Y/Z – Absolute x-/y-/z-position
- I/J/K – x-/y-/z-offset of the current tool position to the center of the radius of a circular movement

For each line of code that starts with one of the listed G-commands an action with the corresponding parameters is added to the list of actions of the current tool object. In the G-code G0 and G1 commands can occur once with the consecutive lines only containing parameter changes but no explicit G-command. To deal with, this the method remembers the last called G-command and adds a corresponding action with the new variable values to the list of actions until a new G-command token is found. The created list of tool objects with sub-lists of actions is then used for the TSV generation.

### 6.1.3 Generation of Tool Swept Volume Models

Each tool object refers to a certain tool number. For each tool number a 3D solid model of the corresponding cutting tool has to be stored in the model repository. These models are considered given and have to be rotationally symmetric around their z-axis with the bottom point or surface of the tool sitting in the x-y plane and extending in negative z-direction. The tool models can be built from more than one part, for example, to include the tool holder in the analysis. An exemplary tool model is shown on the left side of Figure 6-5.

Linear sweeping actions are carried out for all G0 and G1 commands. Each linear sweep requires a shape to be swept, called the “current shape”, and a sweep direction vector. The vector is calculated from the start to the destination position of a movement action as defined by the parameters of the action extracted from the NC file. The vector can be two- or three-dimensional, resulting in planar or non-planar sweeps. This accurately reflects the 2.5D machining capabilities that the approach is developed for. For every sweep a further z-rotation of the current shape needs to be calculated, using the same method as described before, to ensure that the plane built by the axial edges, called the “current plane”, is perpendicular to the sweeping direction vector.

For G2 and G3 commands circular sweeping actions are carried out. As input for the sweeping operation the current shape, a rotation axis and a rotation angle are required. The rotation axis is defined by the center point of the circular movement and a vector. In 2.5D machining, circular cutting operations are only carried out in the x-y plane. The vector is, therefore, always the unit z-vector. The direction of the vector, in positive or negative z-direction, is used to handle clockwise (G2) or counterclockwise (G3) rotations. The position of the center point of the rotation is stored as parameter value in the list of actions as extracted from the NC file. The

angle between the vector from the center of the rotation to the current position of the tool ($\nu_{toCur}$) and the vector from the center of the rotation to the destination position is the rotation angle. For a circular sweep, the rotation of the current shape needs to be set such that $\nu_{toCur}$ is perpendicular to the normal of the current plane.

For the first sweeping action of a tool object, the current shape is the initial tool shape and the current plane is the local y-z plane of the tool. After a sweeping operation the tool shape and the current plane in the final position are saved as input for the subsequent action. This way, the actions and swept volumes are connected with the end position of one movement defining the starting position of the next. The composite solid created by each sweeping operation is directly added to a TSV model. The TSV is complete when all sweeping actions for the current tool are carried out and the next tool object is called. This TSV generation process is carried out until all tool objects in the list are covered. One exception to this process is the first action for every tool object, which is always a movement from the tool change position of the machine to the starting position of the first feed motion. Due to the fact, that the NC file is generated with respect to the WP COS, the tool change position given in the file may be incorrect and not match the actual tool change position of the machine. This movement can, therefore, result in unrealistic sweeps that lead to a falsification of the interference test. Thus, the first movement after a tool change is computed to generate the current shape and plane for the next action, but the generated volume of this sweeping operation is not added to the TSV. This is thought to not have a negative effect on the interference testing, as a collision is most likely to occur at the destination position of the movement action, which is also the starting position for the subsequent movement. Thus, a possible collision would also occur in the subsequent action, leading to a detection of the interference.

The TSV generation results in one (or several partial) TSV per tool used. Each TSV is saved and can be used for the interference analysis and as obstacle in the computational design synthesis. Additionally, the name and file path of each TSV model is added to the FD candidate data.

### 6.1.4 Interference Test

To detect if the TSV interferes with the fixture components, which results in an unwanted collision, the positioning and distance of the geometric entities of the TSV in comparison to the jaws are checked. This test is carried out for every TSV and every FD candidate's assembly model. The assembly models and the TSV models are built in reference to the WP and can, therefore, be combined without the need for placement recalculations. As the TSV models are generally more complicated than the FD candidate assembly models, the TSV stays loaded and the assembly models are exchanged for the test. Thus, every assembly model is checked against the first TSV, then against the second TSV, and so on. In case an interference is detected, this information is added to the FD candidate data.

The interference test is based on a comparison of two shapes. The TSV is first checked against the slide jaw and then against the anvil jaw. Both shapes in the comparison are decomposed into a map of vertices, edges and faces. It is checked in a loop where the vertices of the TSV lie in reference to the solid jaw model. If the vertices are inside the solid model of the jaw an interference is detected. In most cases this test is sufficient to detect a collision. As an additional

test, and only if no interference is found in the first test, a sequence of bounding boxes for the maps of vertices, edges and faces of the two shapes is built and the minimum distance of the bounding boxes is computed for every possible combination of geometric entities: vertex-vertex, vertex-edge, vertex-face, edge-edge, edge-face and face-face. If the resulting minimum of any calculation is smaller than a predefined limit, which is set to 1e-07 mm by default but can be changed by the user, it is considered an interference. This way, contacts or minimal remaining gaps between the TSV and the jaws can also be detected and filed as interference. Although this is not per se correct, it is favorable for the safety of the approach, as in reality the WP locating is subject to the inaccuracies of the fixture device and the stock part itself and the tool locating is subject to the inaccuracies of the machine. These inaccuracies could lead to an interference of the cutting tools and the fixture, even if the 3D models do not collide.

The result of the interference test is added to the FD candidate data marking whether no interference, an interference with one of the jaws or with both jaws was found. In case of an error during the computation this information can also be added to increase the process safety.

### 6.1.5 Verification of Interference Analysis

For the verification of the approach a software prototype is implemented, as presented in a study carried out as part of this work (FLOHR 2014). For each FD candidate the implementation requires a text file as input that contains the file paths to the two vise jaws and the WP model in the *.step format. Using this implementation, the interference analysis is done for several of the case-study parts shown in Appendix 10.2 and manually defined FD candidates for these parts, built from vise jaws created with the design synthesis approach presented in Section 5. The results of the interference analyses are verified by visually analyzing the assembly models of the FD candidates with the TSV models. The NC files for the case-study setups are created ahead of the fixture design, using the machine tool and cutting tool data of the CogMaSh milling machine that the flexible vise is installed on (see Section 1). The NC codes for the studies are given in Appendix 10.8. Additionally, the FD candidate text files containing the locations of the WP and jaw model files are generated manually using the data stored in the AFD ontology.

The 3D solid models of the cutting tools of the milling machine including the tool holders are stored in a sub-directory of the program. A list of these tools with the tool data is presented in Appendix 10.3. The TSV generation is based on existing methods of the used geometric kernel *OpenCascade Technology* (*OCCT*) to sweep the cutting tool models along the paths extracted from the NC file. *OCCT* allows for linear sweeps and rotational sweeps but does not support the sweep of solids (OPEN CASCADE SAS 2013). Therefore, the solid models of the cutting tools need to be transferred into shell representations first. *OCCT* provides methods for this conversion. The resulting shapes consist of shells with edges. The sweeping of edges generates faces and the sweeping of shells generates composite solids. This makes an adjustment to the tool models necessary to ensure the generation of correctly swept models. As shown in the left side of Figure 6-5, all circular axial shells of the tool models are divided in half by two edges that sit in an imaginary plane. If this plane is not perpendicular to the sweeping direction, the edges generate faces that intersect the circular shells of the tool models when being swept. This results in faulty models, as depicted on the right of Figure 6-5.

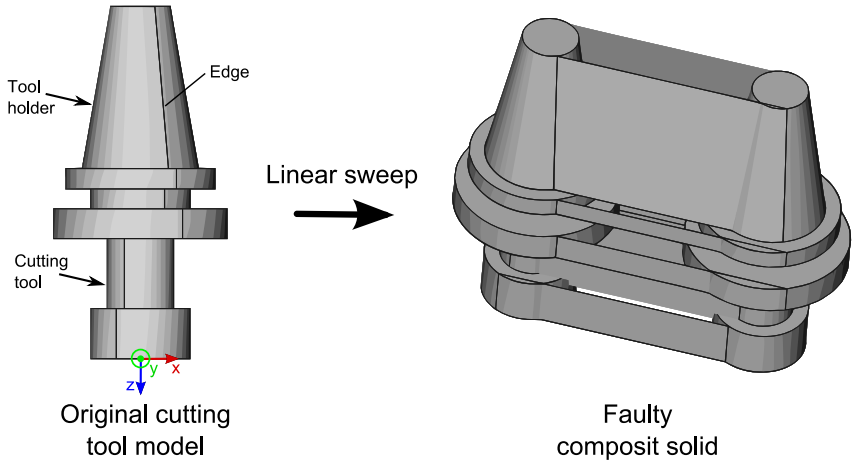



**Figure 6-5 Exemplary original tool model consisting of cutting tool and tool holder (left) and faulty composite solid resulting from linear sweep along the x-axis (right)**

To circumvent this issue, every part of a tool model needs to be rotated such that all axial edges sit in a plane perpendicular to the sweeping direction. To align the edges initially, the parts are rotated such that the edges are in the y-z plane of the tool model. To compute the rotation angles for every part of the tool model, the edges that have vertices with different z-values are identified.

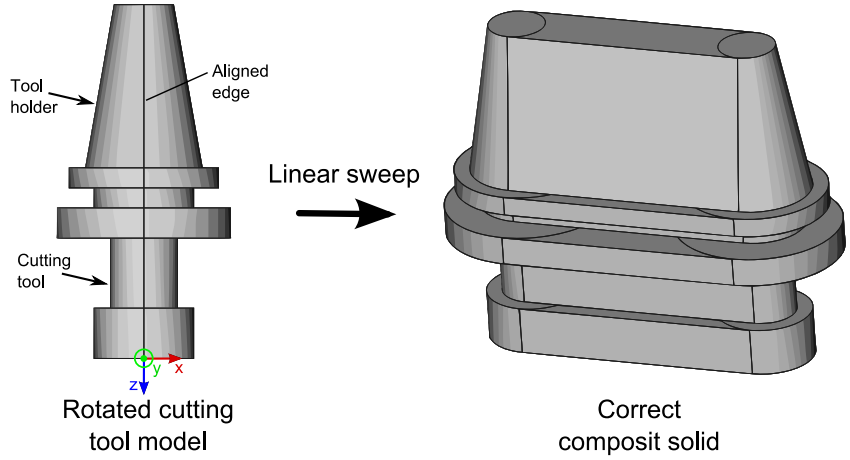



**Figure 6-6 Tool model with rotated parts and axial edges aligned in the y-z plane (left) and correct composite solid resulting from linear sweep along the x-axis (right)**

From this list of edges two are chosen that have equal z-values of a limiting vertex but no joint vertex. Next, a vector from the vertex of the first edge to the vertex of the second edge is created. The angle $\alpha$ of this vector to the normal of the y-z plane of the tool is calculated. The part is then rotated such that the angle $\alpha$ equals $\pi/2$. This is done for every part of the tool model,

resulting in a model where all axial edges are aligned in the y-z plane, as shown in the left side of Figure 6-6. When sweeping this model in a direction perpendicular to the plane the resulting composite shell is correct as illustrated on the right side of Figure 6-6. Once the tool model parts are rotated, the tool model is saved as a single part called the “initial tool model”.

Although supported by the method, only the cutting tools themselves but not the tool holders are used for the generation of swept volumes in the verification studies. This results in less complicated TSV models and reduces the required computation time while still allowing for a verification of the method. However, as the tool holder can have a bigger diameter than the cutting tools, collisions with the tool holder may occur for deep cutouts. These collisions are not determined with this setting. In an industrial scenario, the tool holder and other machine parts that move with the tool should be included in the TSV generation to increase the quality of the analysis results.

To facilitate the application of the program, a simple GUI was implemented using the *Qt* framework (QT PROJECT HOSTING 2014). The GUI allows a user to pick the NC file of the setup in a selection dialog. For automatic processing, the text files describing the FD candidates need to be in the same directory as the NC file. After picking the NC file, the interference analysis process, including the assembly model generation, G-code interpretation, TSV generation and interference test is automatically carried out. The GUI displays the file path of the NC file and all data added to the text files as results of the application. The GUI and the display of results for one example case are shown in Figure 6-7.

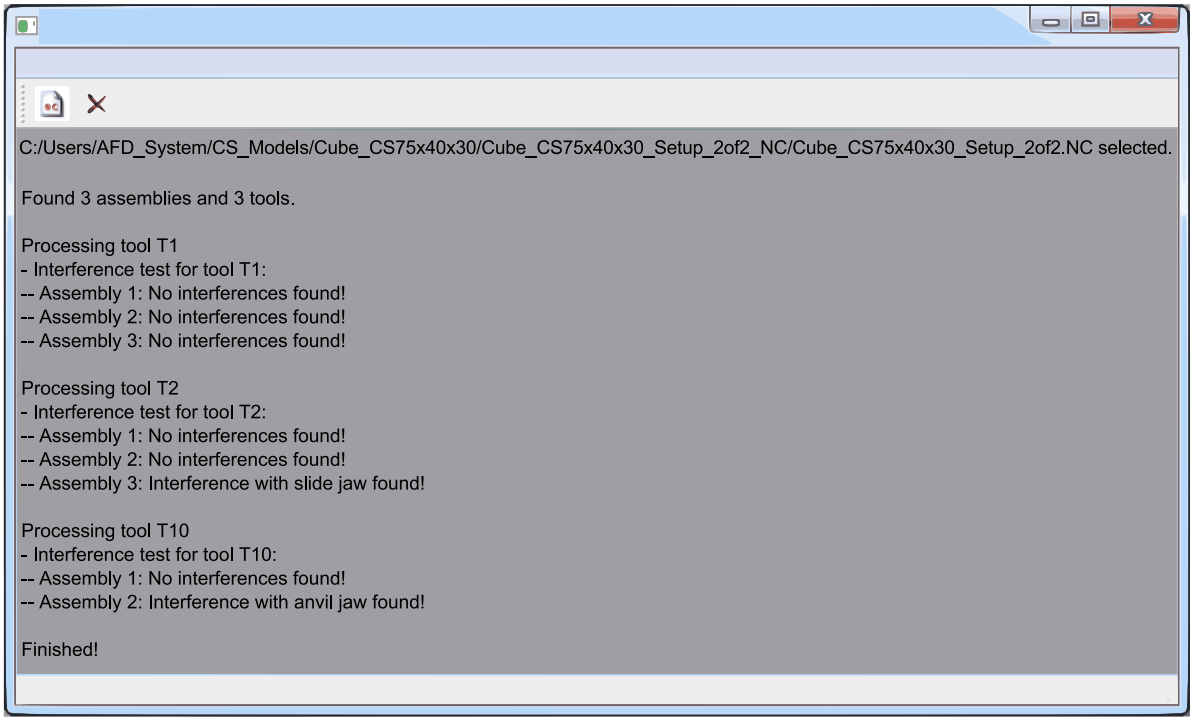



**Figure 6-7 Graphical User Interface of the interference analysis program**

In the following some selected examples from the case-study setups are presented in detail, including the generated assembly models, TSV models and interference analysis results. The chosen examples cover all basic jaw set designs, all tool movement actions and corresponding G-commands and several different cutting tools. Thus, the examples allow for a comprehensive verification of the approach and program.

The first test case is the third and final setup of the part “Winged_Flange”. The stock and the resulting part of the WP can be seen in Figure 6-8 (top). The WP has to be clamped with a step-rest & step jaw set. Two FD candidates with different anvil and slide jaws are proposed. The interpretation of the G-code returns that only the tool T2 is used for the machining of the setup. The outer contour milling operation requires 36 actions. The generated assembly models of the FD candidates together with the generated TSV are shown in Figure 6-8.

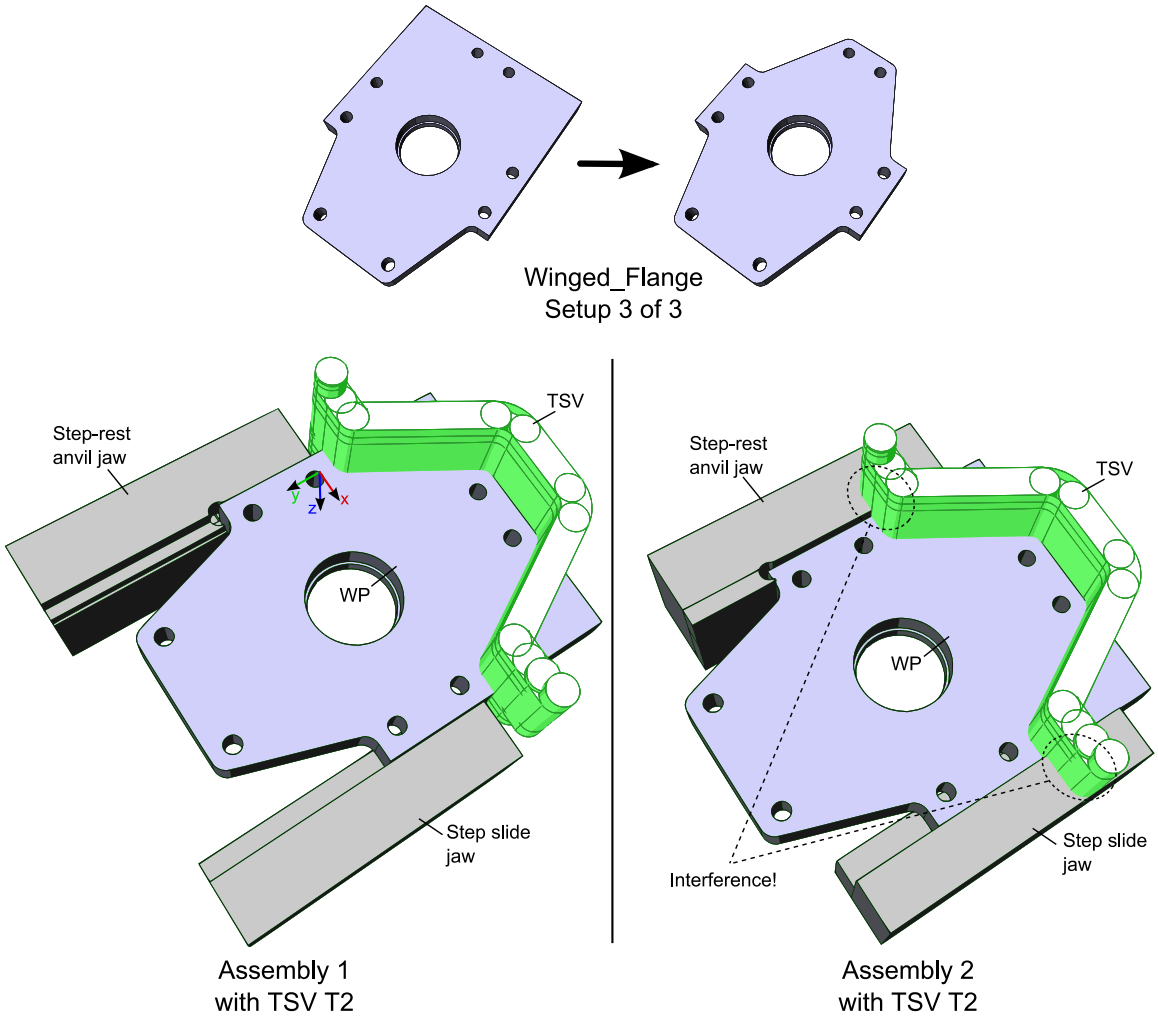



**Figure 6-8 Assembly models and TSV generated by the interference analysis program for the setup Winged_Flange_Setup_3of3 – stock and resulting part (top), one FD candidate with TSV causing no interference (bottom left) and another FD candidate with TSV causing interference at both jaws (bottom right)**

It can be seen that both assembly models are accurately built with the anvil and slide jaws placed correctly in contact with the WP and aligned correctly in space. Further, the TSV is positioned correctly in reference to the WP. The model itself is generated correctly for the most part. An exception to this is discussed in the following section. The program determines no collision for assembly one (Figure 6-8 bottom left) and a collision with both jaws for assembly two (Figure 6-8 bottom right). Considering the depicted combined assembly models these results are correct. In the first assembly, the TSV does not touch any of the jaws. In the second assembly, the tertiary locator of the anvil jaw (locating in y-direction) is placed such that the outmost part of the TSV collides with both jaws. The example shows that the assembly model creation for step-rest & step jaw sets and the TSV generation for linear and circular sweeps works and that the detection of no collisions and collisions between the TSV and both jaw models works.

The next test case is the second and final setup of the WP "Cyl_Plane". This WP, as depicted on the left side of Figure 6-9, requires V & step jaw designs for locating. Again, two FD candidates with different jaw geometries and the NC file for the setup are given. From the G-code, two tool objects are extracted: the face-milling tool T1 is used to remove the base part in 19 actions and the end-milling tool T9 is used to machine the pocket which requires 62 actions. Figure 6-9 shows the first generated assembly model with the WP as wireframe and the TSV for T1 (bottom left) and the TSV for T9 (bottom right).

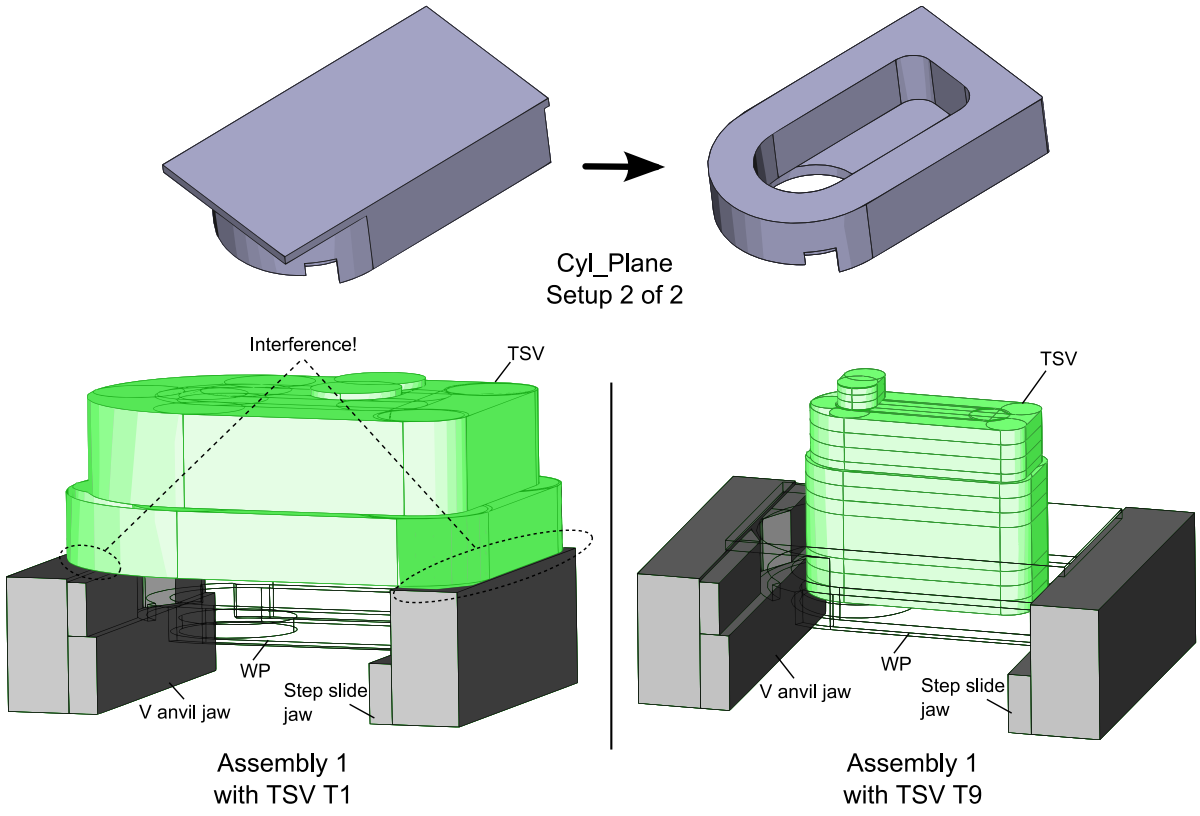



**Figure 6-9 Assembly model and TSV generated by the interference analysis program for the setup Cyl_Plane _Setup_2of2 – stock and resulting part (top), FD candidate with first TSV causing interference (bottom left) and same assembly with second TSV causing no interference (bottom right)**

Again, the assembly models are accurately generated with an exact placement of the V-shaped anvil jaw and the step-shaped slide jaw in contact with the WP. The jaws are also correctly aligned. Both TSV are generated correctly. The interference test determines that for the first assembly the TSV of T1 interferes with both jaws. The tests for the other TSV and the other assembly return no interferences. Thus, the models of the other assembly are not depicted. Considering the model of assembly one and TSV T1, it can be seen that the bottom plane of the tool swept volume touches but does not penetrate the top surface of both jaw models. As previously explained, the test regards this as an interference. The result is, therefore, correct. Same goes for the determination of no further collisions in the other tests. The example shows that the assembly model creation for a V & step jaw set works, that the assembly model generation in case of different supporting surface heights works and that a contact between the TSV and the jaws is correctly determined as an interference, even though the TSV does not penetrate the jaws.

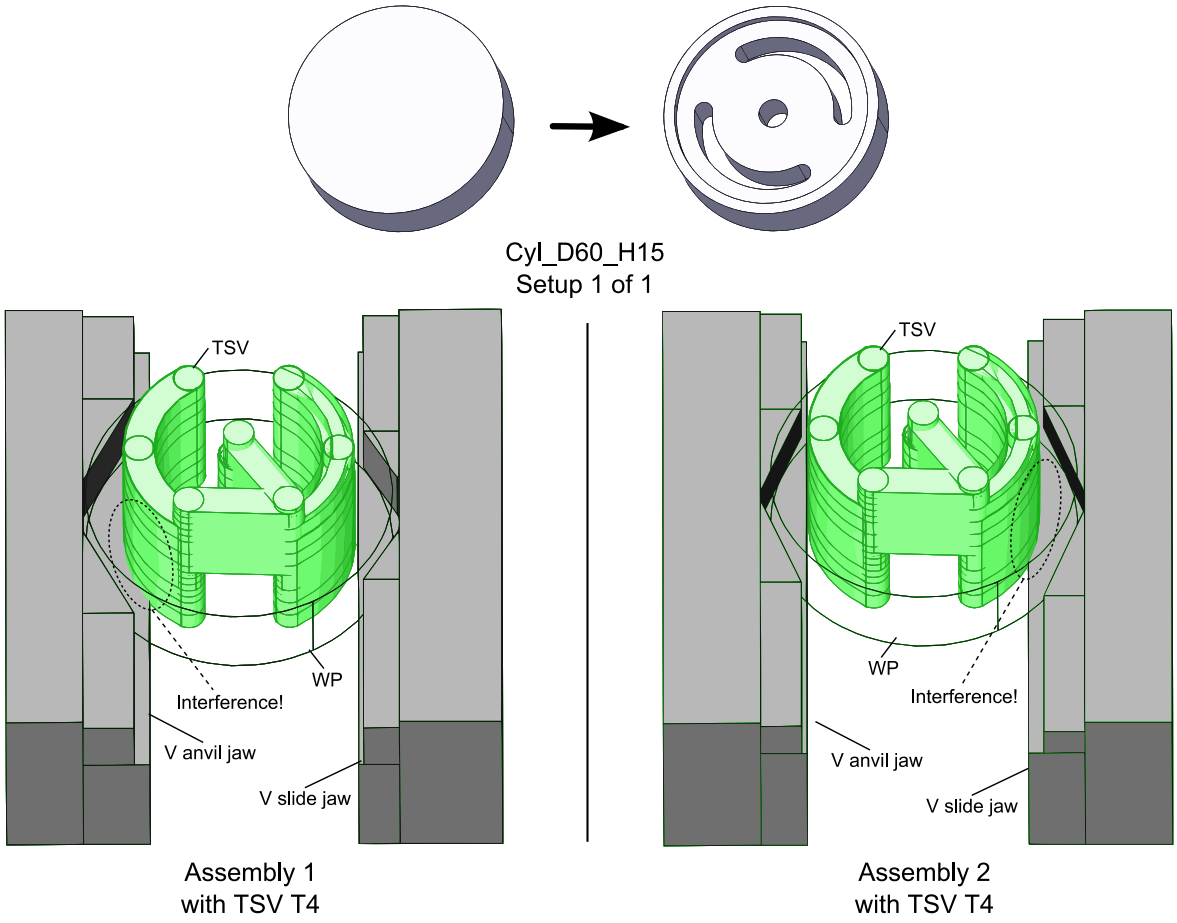



**Figure 6-10 Assembly model and TSV generated by the interference analysis program for the setup Cyl_D60_H15 _Setup_1of1 – stock and resulting part (top), FD candidate with second TSV causing interference at anvil jaw (bottom left) and other FD candidate with second TSV causing interference at slide jaw (bottom right)**

In the final example the only setup of WP “Cyl_D60_H15” is used. The stock and resulting part are shown in Figure 6-10 (top). Two FD candidates using V & V jaw sets are provided. From the NC file it is extracted that the tool T6 is used for a face milling and pocketing operation which requires 80 actions and the end mill T4 is used for the machining of the grooves and hole. This requires 139 actions. In Figure 6-10, the TSV for tool T4 is shown with the first generated assembly model (bottom left) and with the second assembly model (bottom right). In both assemblies the WP is depicted as wireframe.

Upon inspection of the models it can be seen that the system correctly assembles the parts also for jaw sets with V-shaped slide jaws. The TSV are also generated and placed correctly. The TSV of T6 does not cause an interference and is, therefore, not shown in the figure. The TSV of T4, however, collides with one jaw in both assemblies. This is caused by the breakthrough of the milling tool at the bottom WP surface when machining the circular groove in combination with the step depth of the respective jaw. The collisions are correctly determined by the system which returns an interference between T4 and the anvil jaw for assembly one and T4 and the slide jaw for assembly two. This example proves that the assembly model creation also works for V & V jaw sets and that interferences with either the anvil or the slide jaw are also correctly determined.

### 6.1.6 Discussion of the Interference Analysis

As shown in the verification, the interference analysis with the presented approach and implementation can be successfully applied to the case-study parts and setups. The generated TSV and assembly models as well as the results of the interference test are accurate. All further tests with other demo parts and setups shown in Appendix 10.2 were also successful. This includes tests with a higher number of tools or FD candidates. The use of tokens in the NC file interpreter and TSV generation allows for an easy extension of the method and implementation to cover other commands, such as further movement or feed-rate and spindle-speed commands. The use of other tool models is also possible, given that they follow the modeling guidelines presented in Section 6.1.3.

In the presented approach rapid movement actions (G0 commands) are assumed to be linear horizontal or vertical movements in one or two directions. This does not necessarily have to be the case and depends on the settings and capabilities of the machine tool. For the milling machine that the vise is installed on, and as common in 2.5D machining, the G0 commands represent linear 2D movements. In case 3D movements are possible, an extension to the program interpolating the G0 actions with linear movement actions is required. As this increases the computation time and yields no benefit for the flexible vise, this approach was not pursued.

An aspect that is currently not included in the approach is the consideration of WP tolerances, as this was initially excluded from the scope of the thesis (see Section 1.5). However, tolerances on the WP dimensions are important for the interference analysis as they can affect whether a collision occurs or not in case the tolerance value is bigger than the distance limit used for detecting collisions. The user-defined distance limit between geometric entities that is checked and used to determine collisions or contact of entities is, thus, an option to including tolerances in the presented analysis approach. As presented, the limit was set to 1E-07 mm for the case

studies. By using the maximum absolute tolerance value of the WP dimensions as limit, tolerances could be included in the interference check. An automation of this would also require that the tolerances are represented in the 3D models of the WP, e.g. as annotations, or are stored in and retrieved from the AFD ontology.

An unresolved issue that occasionally occurred is the incorrect generation of shells resulting from circular sweeps. Shells of the current shape that are parallel to the x-y plane are sometimes not correctly generated leading to gaps in the models. The error is caused by the revolve sweep method used for the circular sweeps. However, despite extensive testing, no reason for or regularity of this error could be determined. The gap sometimes does affect the subsequent action, which can lead to missing shell elements even in linear sweeps. An example of this is shown in Figure 6-11. On the left side of the figure, the resulting volume from a single circular sweep of a cutting tool is shown. This volume has a missing top surface, which is parallel to the x-y plane. The error can also affect the bottom surface, as this is also in the x-y plane. When such an error occurs and the next action is a linear sweep, the model resulting from that sweep can be, but does not have to be, erroneous, too. This is shown on the right side of the figure. However, as the shell elements that are not in the x-y plane are generated correctly, the edges and vertices of the shell element are still present in the TSV as part of the adjacent shells. Due to the interference test method, which not only checks for penetrating shells but also uses a distance calculation between vertices and edges, collisions are correctly determined even if shell elements in the x-y plane are missing. The missing shells, therefore, do not lead to a falsification of the interference analysis but may have a negative effect when the TSV models are used as obstacles in the computational design synthesis approach presented in Section 5.

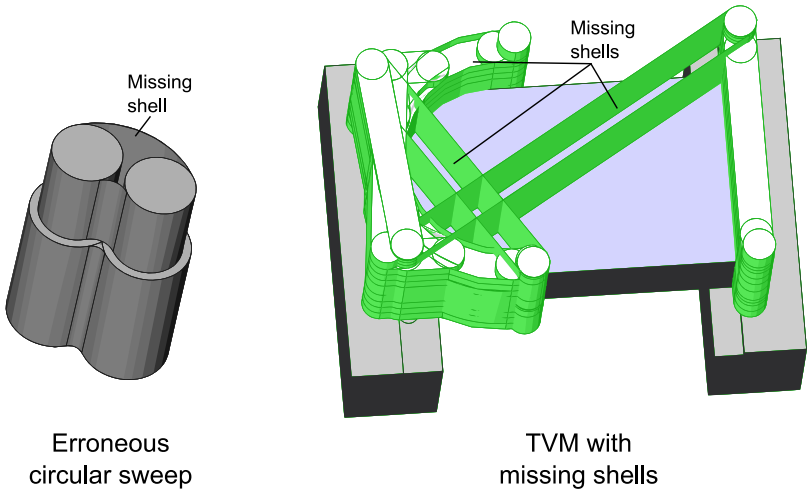



**Figure 6-11 Missing shells in x-y plane resulting from circular sweep (left) and from linear sweeps as follow up to erroneous cylindrical sweep (right)**

When testing the implementation of the presented approach, a memory management issue occurred in *OCCT* when the amount of sweeping actions for a tool is very high. This memory management issue causes an abortion of the TSV generation process. As the machining of

complex parts often requires a high amount of movement actions of a single tool, a solution for this problem needs to be found. Every sweep generates a number of elements that directly depends on the number of elements of the swept object, i.e. the shell representation of the tool. The generated TSV can, therefore, have a very high amount of shell elements (and edges and vertices). When inspecting the TSV it can be seen that many of these geometric elements are not part of the outer contour of the TSV but lie inside the volume. For the interference analysis, however, only the outer contour is important. All elements that are within the TSV could be deleted to reduce the model complexity. This process is known as "trimming" of a swept model and is recognized as one of the most difficult aspects in the generation of swept volumes (BLACKMORE et al. 1999). The method basically allows for the inclusion of trimming methods in different stages of the TSV generation process. Several approaches to trim the TSV were, however, tested without any success. Attempts to re-compute only the outer shell of the TSV after each added swept shape or for the complete TSV fail as unsatisfactory or even no results are generated. Removing overlapping internal shells and edges using a Boolean cut or Boolean union and subsequent shell operation after every added swept shape also causes faulty models. An approach to delete all internal shells and edges of the final TSV leads to incorrect models or a very high computation time. The complexity of the shapes in combination with the somewhat limited capabilities of the methods provided by *OCCT* is assumed to be the reason for the occurring errors. These limitations are known to the developers of *OCCT* and are actively being worked on. Future versions may, thus, resolve this issue. As a workaround, the program splits a TSV into several models if the list of actions exceeds a pre-defined number. Hence, each partial TSV contains the shell elements generated by the defined number of actions. The setting of this number depends on the complexity of the tool model, which affects the number of shells added to the TSV for each action, and needs to be determined heuristically.

Although this work-around allows for the successful application of the approach and implementation, the remaining high model complexity negatively affects several aspects: To simplify the testing, reduce the number of generated models and especially to have a single model that can be used as an obstacle in the design synthesis process of new vise jaws, the volume models of all tools are to be combined into one total TSV (TTSV). By merging the existing TSV models in one file, the generation of a TTSV is generally possible. However, because the model still has all the details of the original shell models, the files are of big size and cause very high computation times in the design synthesis application process. Considering the interference test method, the missing trimming affects both the selection of the method and its performance. The distance measure is computationally expensive as the high number of shells leads to a high amount of geometric entities that need to be compared against each other. The use of a trimmed TTSV would lead to a drastic saving in computation time. Further, the use of a TTSV solid would allow for the use of another method, the Boolean intersection between the TTSV and the fixture components, to detect interferences. Testing this approach with manually created solid TTSV showed a further saving in computation time compared to the distance measure method. To allow for a realistic interference analysis, the tool holders should further be included in the TSV generation. The approach allows for this, but including the tool holders causes a higher number of geometric entities and, thus, leads to memory management problems faster. If a trimming method is added to the approach, tool holders and other peripherals can be included in the tool model.

In the presented implementation, the tasks of generating the TSV and running the interference analysis are carried out in one automated sequence. To use the TSV as obstacle(s) in the computational design synthesis process, however, the TSV generation has to happen independently before the fixture configuration and design verification phase. Due to the object oriented programming used, a split into stand-alone modules for G-code interpretation, TSV generation, assembly model generation and interference analysis is realizable with little effort.

Considering the degree of automation, the approach requires no user interaction per se but the presence of certain input data, such as the WP and jaw models for the assembly model generation, the tool models and the NC file to be processed. The implementation uses one text file per FD candidate that states the file paths of the models required for the assembly model. The text file is also used to store the test results as string value when the analysis is carried out. The file paths required as input are stored in the AFD ontology and the analysis result can be stored by setting a Boolean property value of the candidate to true or false. This is sufficient to filter out infeasible candidates for subsequent fixture design process steps. The interference analysis implementation further requires a certain folder structure and naming of the files to work. This structure can be changed if needed or, if more user interaction is required, a selection dialog can be included using existing *Qt* methods. These alterations were tested successfully. The program can be controlled via the GUI, where only the NC file has to be selected by a user to start the analysis process. Additionally, a command line start, i.e. a start without the GUI, is possible in which the location of the NC file has to be given.

## 6.2 Workpiece Deformation Analysis using FEM

The second type of analysis that is used to evaluate the FD candidates is the deformation analysis based on the finite element method. The aim of the analysis is to determine the FD candidate among the interference-free candidates that leads to a minimum deformation of the WP when it is subject to the clamping and machining forces. The analysis is based on the assembly models of the FD candidates as generated by the spatial grammar or the assembly model generator that was presented as part of the interference analysis.

A basic structure of the FEA process is depicted in Figure 6-12. The clustering of the tasks shown in the algorithm is based on the software tools used (*Salome Meca*), the input they require and the output they generate. However, the basic process is the same for all FEA tools. The main process steps are discussed in detail in the following sections.

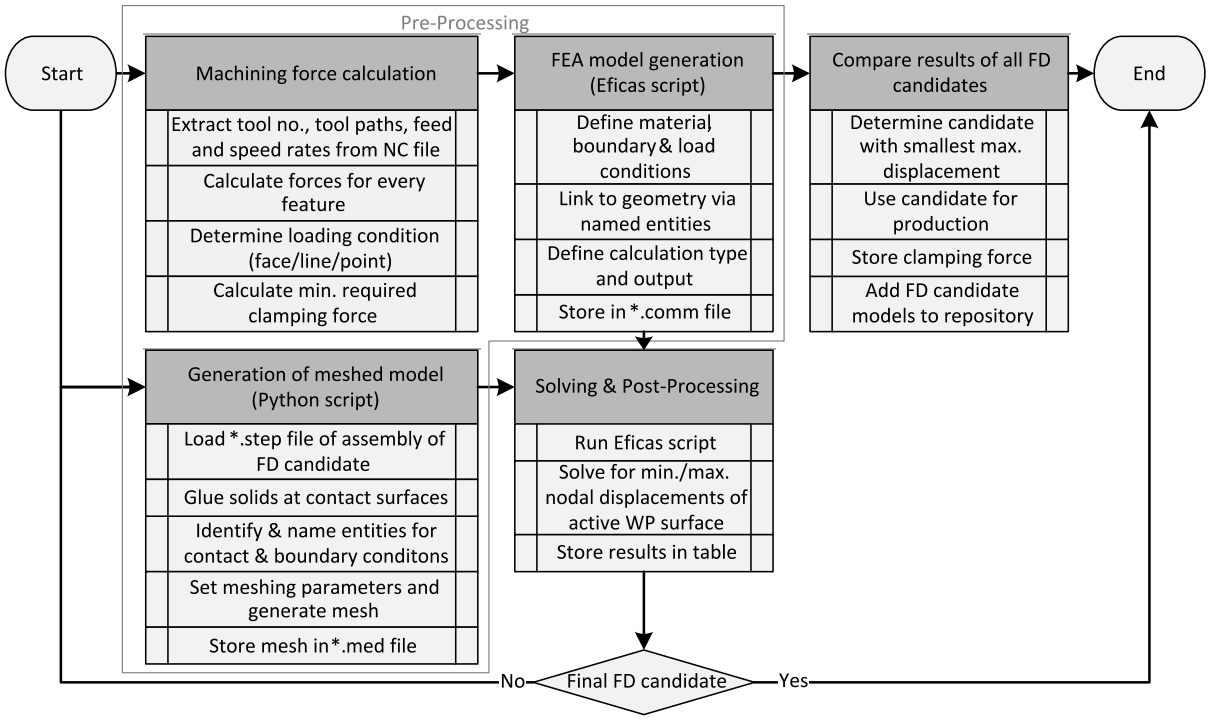



**Figure 6-12 Basic algorithm of the finite element analysis for the fixture design candidates**

## 6.2.1 Machining and Clamping Force Calculation

Both the clamping and the machining forces, i.e. the forces that the cutting tools of the milling machine induce upon the WP during any milling or drilling process, can add to the deformation of the WP. Therefore, it is essential to include these forces in the analyses as loads (AMARAL et al. 2005). An exact modelling and calculation of the machining forces is a non-trivial task as it consists of several forces that are dynamically influenced by various factors. For horizontal end-, face- or groove-milling, a common calculation approach splits the machining force into three parts, the passive force in axial direction, the feeding force in tool feed direction and the cutting force perpendicular to it and tangential to the cylindrical tool surface. In case of a horizontal tool movement the two latter force components are horizontal, too. For vertical drilling, the force is split into a horizontal cutting force and a vertical feeding force and passive force (PEROVIĆ 2013, p. 33ff).

Although machining plans in general are to be generated such that the cutting tools work against the locating and supporting contacts, the milling tools may move in different directions during the machining process. The clamping forces have to withstand the machining forces at all times during the machining process, to guarantee force closure of the WP and, thus, a stable positioning of the WP in the reference position. To ensure a secure operation of the fixture, the worst-case scenario, i.e. the maximum machining forces working directly against the slide, i.e. against the clamping force, or in the direction of an unrestricted translational degree of freedom (DoF) of the WP, i.e. against the reaction forces caused by the clamping force, has to be

assumed (KASHYAP & DEVRIES 1999). Further, the following assumptions are made that simplify the calculation process. The assumptions all lead to conservative results and are based on common cutting force modeling approaches presented in literature (DEGNER et al. 2009, pp. 90-105).

- The passive cutting force components are generally not considered, as they are very small in comparison to the cutting and feeding force.
- The tool entry and exit phase, which would cause a sinusoidal cutting force gradient, is not considered. Instead, the milling and drilling tools are assumed to always cut into solid material with their full diameter (drilling) or half diameter (horizontal milling). This causes the maximum force values.
- For the friction force calculation, the clamping force is assumed to act only in negative x-direction.

Based on these assumptions, the machining forces are calculated using Kienzle's formulas as presented by PEROVIĆ (2013, p. 33ff). The formulas require machining data as well as geometric and material data from the tool and the WP. The machining data, such as feed rates and spindle speeds, are considered given and can be retrieved from the existing NC files of the WP. The geometric data, such as cutting depth and width, can also be retrieved from the NC files by analyzing the tool paths. The necessary material-specific data for both the tool and the WP are retrieved from FISCHER et al. (2011) as these specific values are not part of the material ontology used in the AFD ontology (see Section 3.2). The geometric data of the cutting tools, like the diameter, the number of teeth or the tool rake angle, is stored in the tool table as presented in Appendix 10.3. With this data, the cutting and feeding force is calculated for all cutting operations (milling and drilling). As opposed to the full Kienzle formulas simplified denotations are being used that do not take the detailed correctional values for tool wear, tool material and cutting speed into account but use a single correctional factor K, instead. The components of the formulas are defined in Table 6-2.

**Table 6-2 Components of the machining force calculation formulas (2)-(7)**

| Abbr. | Description | Abbr. | Description |
|---|---|---|---|
| 1-mc | Cutting force exponent | κr | Tool rake angle |
| 1-mf | Feeding force exponent | z | No. of teeth |
| kc1.1 | Specific cutting force | φs | Tool enclosure angle |
| kf1.1 | Specific feeding force | n | Spindle speed |
| ap | Cutting depth | f | Feed rate |
| ac | Cutting width | K | Correctional value |
| D | Tool diameter | | |

The **milling cutting force $F_c$** is calculated using equation (2), and the corresponding **milling feeding force $F_f$**, in case of peripheral milling, is calculated using equation (3), both with the average chip-width **$h_m$** defined as denoted in equation (4):

$$F_c = \frac{a_p}{sin\kappa_r} * h_m^{1-m_c} * k_{c1.1} * z_{iE} * K \quad (2)$$

$$F_f = F_c * cos\varphi_s = -F_c \quad (3)$$

$$h_m = \frac{360}{\pi * \varphi_s} * \frac{a_c}{D} * sin\kappa_r * \frac{f}{n * z} \quad (4)$$

For **drilling** operations the **cutting force $F_{cz}$** per tooth can be calculated by equation (5) and the **drilling feeding force $F_f$** with equation (6). The average chip-width **$h_d$** for drilling into solid material is calculated using equation (7):

$$F_{cz} = \frac{D}{2 * sin\kappa_r} * h_d^{1-m_c} * k_{c1.1} * K \quad (5)$$

$$F_f = D * h_d^{1-m_f} * k_{f1.1} * \mathrm{K} \quad (6)$$

$$h_d = sin\kappa_r * \frac{f}{n * z} \quad (7)$$

The resulting machining forces are used to determine the necessary clamping force via a force equilibrium condition. If the maximum resulting horizontal force acts directly against the slide, the minimum clamping force has to be at least equal to it, to ensure that the slide is not moved during machining. To guarantee a secure operation a safety factor S can be included, as shown in equation (8).

$$F_{Clamp,\,min,\,dir} = F_{mach,horizont,max} * S \quad (8)$$

Additionally, in case of a Step-rest-shaped anvil jaw and the resulting unrestricted DoF in (positive or negative) y-direction, the clamping force that leads to sufficient frictional forces between the WP and the jaws has to be calculated. The frictional force is calculated with the following basic formula:

$$F_{friction} = \mu_{static} * F_{normal} \quad (9)$$

As the WP should stay in a static position, the static frictional coefficients are to be used for the calculation. The normal force that causes the frictional forces is the clamping force $F_{clamp}$. As a simplification, material damping and energy dissipation through part deformation is neglected. Thus, the full clamping force acts at both the anvil and the slide jaw contact surfaces. In the equilibrium case, the sum of both resulting frictional forces $F_{friction}$ has to (at least) equal the maximum horizontal machining force $F_{mach,\,max}$.

$$F_{mach,\,max} = F_{friction,Anvil} + F_{friction,Slide} \quad (10)$$

with:

$$F_{friction,Anvil} = F_{friction,Slide} = \mu_{static}\ F_{clamp,\ min} \quad (11)$$

follows:

$$F_{clamp,min,frict} = \frac{F_{cut,max}}{2*\mu_{static}} *S \quad (12)$$

In the first case, where the machining force works directly against the slide the minimum clamping force has to equal the maximum cutting force. For the second case, the formula for $F_{clamp,min}$ shows that when $\mu_{static}$ , the static friction coefficient between the jaws and the WP, is less than 0.5, the minimum clamping force has to be even bigger than the maximum clamping force. The higher minimum clamping force value of the two has to be used to ensure WP stability in the fixture, i.e. that the clamping and contact reaction forces withstand the machining forces. For this clamping force value, the resulting WP deformation has to be analyzed next. Again, a safety factor S can be included in the calculation.

## 6.2.2 FEA Model

The general FEA model is designed to enable the retrieval of qualitative results, i.e. the detection of the FD candidate that causes the minimum WP deformation. Achieving accurate quantitative results is of less interest for an evaluation. Further, the FEA model should be generally applicable to all possible jaw set and WP shapes and allow for automation of the analysis. Especially if a high number of design candidates have to be analyzed, the calculation time can add up significantly. Thus, the calculation time per analysis should be kept low.

These premises lead to preference of a linear analysis with glued contact surfaces in comparison to a non-linear analysis with frictional contacts between the jaws and the WP because the linear analysis drastically shortens the calculation time and helps to avoid the problem of non-converging solutions. The effect of not considering the frictional contact is discussed later.

### Geometric model and meshing

For the analysis, the base construction of the fixture, i.e. the vise rail, anvil and slide, is not included to simplify the geometry and reduce the computing effort and time. As the base construction is laid out to withstand much higher forces than the achievable clamping forces it is assumed rigid. In a study on the accuracy of FEA results for fixture-WP deformation analysis SIEBENALER & MELKOTE (2006) found out that in 98% of the tested cases only modelling the WP and the fixture components, which are in contact with the WP, leads to the same accuracy of the results (with a maximum deviation of 5%) than the inclusion of the whole fixture device in the FEA, given that accurate boundary conditions are defined to represent the fixture body. The computing time, on the other hand, drastically increases when the fixture body is included. Hence, only the fixture contact components, which are the vise jaws, are included in the FEA.

The assembly models of the WP and jaw set of each FD candidate, as generated by the spatial grammar or by the assembly model generator presented in the previous sections, are used for the analysis. This allows for the processing of FD candidates generated by ontological reasoning and by computational design synthesis. The models are meshed using the auto-meshing function of *Salome*. For the automatic meshing function to work stably, a 3D 4-node

tetrahedral element type is used. Although not as accurate as hexagonal elements, 3D tetrahedral 4-node elements have been found suitable for FE-analyses in fixture design (KASHYAP & DEVRIES 1999).

An initial average and maximum element size for the tetrahedral elements has to be defined. This is done via the maximum line length of the elements, which can be used to control the average element size and number. As common with the FE method, the element size is a trade-off between result accuracy and calculation time and effort. A finer mesh leads to a higher number of nodes and to more accurate results but also to a higher calculation time.

### Boundary conditions, loads, material models

Once the geometry is meshed, the boundary conditions can be applied. To represent the effects of the base construction of the fixture, the displacement of the backside of the anvil jaw, which is in contact with the anvil, is completely constrained in all translational directions. Further, the displacement of the backside of the slide jaw is constrained to only allow for a movement in the translational direction of the slide. This represents the assumption that the jaws are "glued" to the anvil and slide at the contact surfaces during the whole machining process, that the base construction is rigid and that the slide can only move in x-direction, which is the direction of the pneumatic piston that drives the slide and exerts the clamping force. Considering the high stiffness of the fixture device and the form- and force-closure established between the jaws and the slide and anvil of the vise this assumption is considered valid. No further constraints are implied as all other surfaces, including the bottom surfaces of the jaws, are not subject to any contact or restriction. Figure 6-13 shows the displacement boundary conditions and the clamping force application.

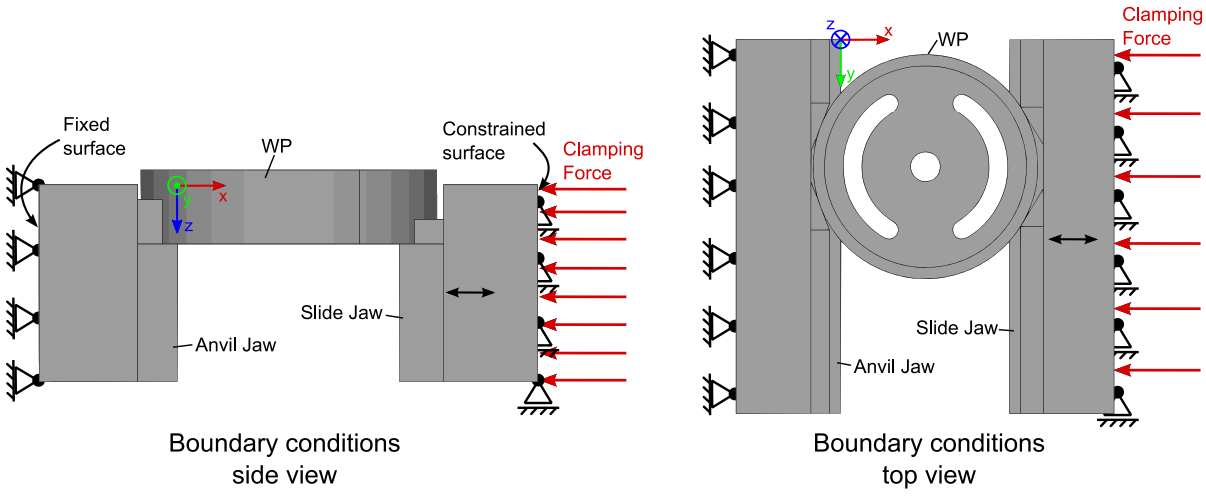



**Figure 6-13 FEA boundary conditions on anvil and slide jaw representing the rigid fixture body and the clamping force applied as surface load**

As loads, the clamping and machining forces are applied. Gravity is not considered in the analysis as it has a minimal influence on the WP deformation. The clamping force is applied in

negative x-direction to the back surface of the slide jaw as a surface pressure load. This accurately represents the force distribution that the slide exerts on the slide jaw. Further, the machining forces are applied to the WP in the form of face, edge or nodal loads, depending on the type of machining operation performed. For drilling operations, the feeding force is applied as a nodal force in the positive z-direction distributed along the nodes of the top edge of the borehole. The cutting force is applied at two opposing nodes of the same edge in opposite directions to model the torque induced by the tool. Using nodal loads leads to slightly larger displacements and higher local stress at the respective nodes. However, as the stress is not of interest in the analysis at hand and larger displacements are conservative, this assumption is valid (AMARAL et al. 2005).

For face milling operations, the cutting and feeding forces are applied as surface load to the surface of the feature being milled. For groove milling operations, the forces are applied as edge loads. Despite the actual tool path and the resulting force directions, the forces are applied in the x- and y-direction such that they point against the anvil jaw locating surfaces, as this state causes the maximum WP deformation. The analysis is carried out consecutively for every machining operation, i.e. for every feature, always taking the clamping force and displacement boundary conditions into account.

A linear, isotropic elastic material model is used for the jaws and the WP. This resembles most engineering grade materials that are to be machined with cutting tools and meets the requirements of the manufacturing scenario the flexible vise is used in, where only aluminum, plastics and some steel-type materials are processed.

By this all necessary settings to solve the analysis problem for the WP deformation are described. To evaluate the approach and the FE model, the analysis is carried out for some of the case-study parts as presented in the following.

### 6.2.3 FEA Evaluation Studies

Two of the case-study setups shown in Appendix 10.2, Winged_Flange_Setup1of3 and Cyl_D60_H15_Setup1of1, are used to evaluate the FE approach. For every setup, three different FD candidates are processed. The six resulting studies cover all possible jaw set shapes and all sort of milling and drilling operations respected in the approach. Figure 6-14 shows one candidate for each setup. The complete list is shown in Appendix 10.9. In the following, the study names will be shortened WF for the Winged_Flange_Setup1of3 and CYL for Cyl_D60_H15_Setup1of1.

The analyses are carried out using the *Salome-Meca* software in an interactive fashion, i.e. based on user input. The aspects of automating the analysis using this tool (set) are discussed subsequently.

The vise jaws and the WP are made out of 6061 T6 aluminum. The respective material data (E-modulus and Poisson's ratio) is retrieved form the AFD ontology and used in the analysis. Further, a linear elastic isotropic material model is defined. The displacement boundary conditions and the clamping force, described in Figure 6-13, are defined for the respective geometric entities. As the clamping force is applied as surface pressure, the force value has to be divided by the surface area, which is constant.

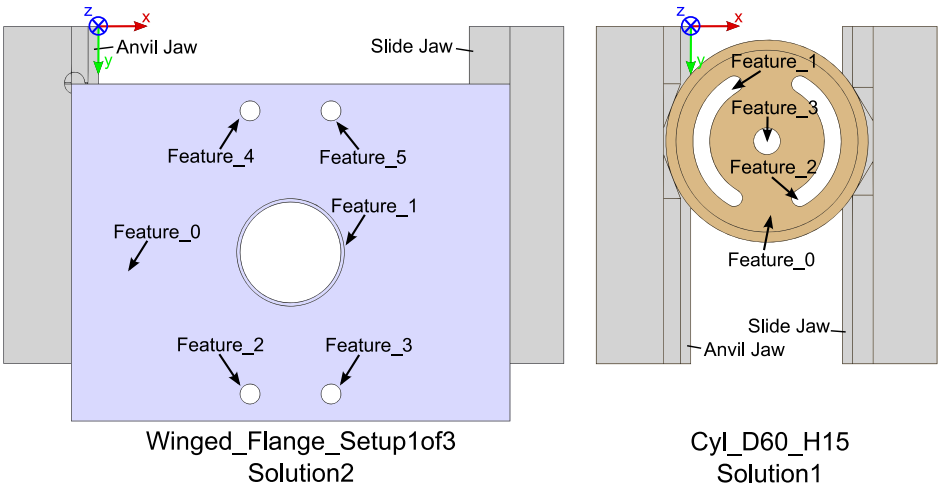



**Figure 6-14 One FD candidate for each case-study setup used for the evaluation of the FEA approach – Winged_Flange_Setup1of3 (left) and Cyl_D60_H15_Setup1of1 (right)**

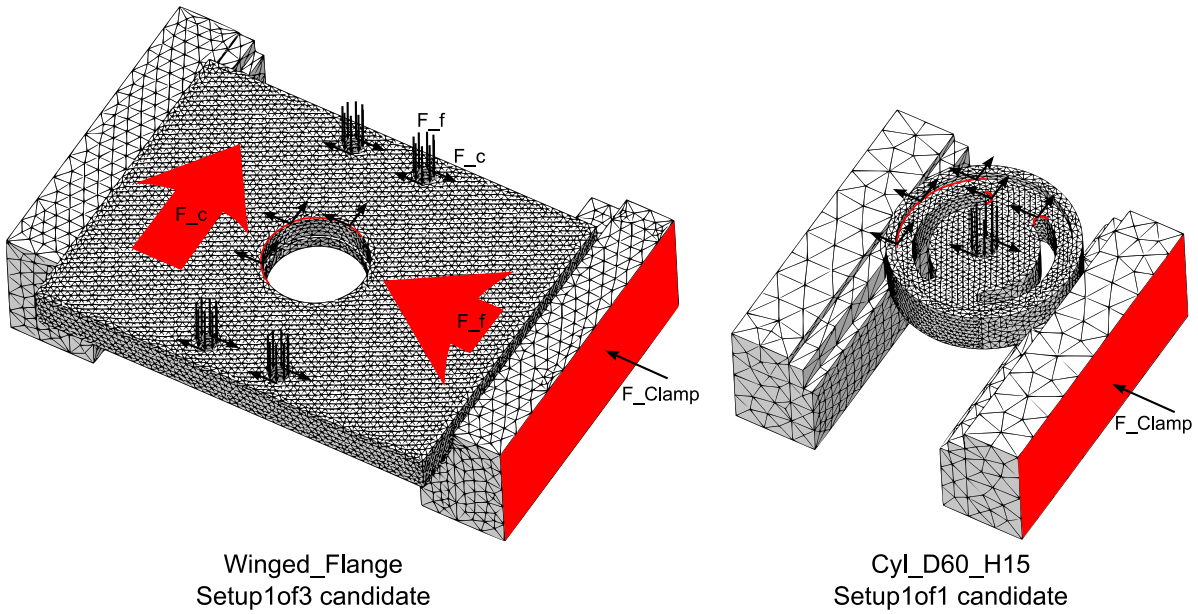



**Figure 6-15 Machining force application as surface, edge or nodal loads for the different features of Winged_Flange_Setup1of3 (left) and Cyl_D60_H15_Setup1of1 (right)**

Next, the machining load for each feature is defined and applied. As shown in the left side of Figure 6-14, six features are machined in the WF setup. Feature_0 is a face milling operation where the machining forces are applied as a face load. The machining forces of Feature_1, the

circular pocket, are applied as an edge load on one half of the circle on the top of the WP. Features two to five are the bore holes that are being drilled. Here, the load is applied as several nodal forces. On the right side of Figure 6-14 the four features that are to be machined for the CYL setup are highlighted: a circular milled pocket (Feature_0), two grooves (Feature_1 and _2) and a drilled central hole (Feature_3). Like in the previous study, the machining forces are applied as edge or nodal forces depending on the feature type. The force directions are chosen to work against the locating entities of the anvil jaw and do respect the turning direction of the tool as described above. Figure 6-15 depicts the meshed models with boundary and loading conditions for both studies.

Using these loads and boundary conditions, the analyses are carried out and solved for nodal displacements of the overall assembly. Additionally the minimum and maximum displacement values for every translational DoF of the nodes of the active WP surface are stored as a table.

### Machining and clamping force calculation

Using formulas (2) to (7), the feeding and cutting forces for the machining of every feature of the two setups are calculated. The necessary input data is listed in Appendix 10.10. The tools used to machine a feature are derived from the machining plan (NC file) of the respective setup. In the given studies, every feature is machined with one tool and constant machining parameters. If, however, multiple tools or different parameters are required for one feature the maximum occurring machining forces are to be used. The moments generated are not calculated directly but modeled as opposing forces in the analysis.

Based on the machining forces, the minimum required clamping force can be calculated using equations (8) & (12). In both studies the Jaw-WP material pairing is aluminum with aluminum which results in a static friction coefficient of $\mu_{static}$=1.05 (FISCHER et al. 2011). Hence, the minimum clamping force is defined by equation (8). Using a safety factor of S=2, the clamping forces for both studies are calculated. The results of the force calculations are shown in Table 6-3.

**Table 6-3 Absolute values of the machining forces calculated for every feature of the study parts**

| Case-Study | Feature_ 0 | Feature_ 1 | Feature_ 2 | Feature_ 3 | Feature_ 4 | Feature_ 5 | F_Clamp |
|---|---|---|---|---|---|---|---|
| **Winged_ Flange Setup1of3** | Fc: 1525N<br>Ff: 1525N | Fc: 1740N<br>Ff: 1740N | Fc: 1281N<br>Ff: 266N | Fc: 1281N<br>Ff: 266N | Fc: 1281N<br>Ff: 266N | Fc: 1281N<br>Ff: 266N | $F_{cl}$: 3480N |
| **Cyl_ D60_H15 Setup1of1** | Fc: 1740N<br>Ff: 1740N | Fc: 1509N<br>Ff: 1509N | Fc: 1509N<br>Ff: 1509N | Fc: 2033N<br>Ff: 924N | -- | -- | $F_{cl}$: 4066N |

## Results

The calculated clamping and machining force values are applied to the respective analyses. To determine the influences of the mesh granularity, the critical displacements for the WP deformation and to verify the basic FEA model, some initial analyses based on the study WF_Candidate2 are carried out first. This study and candidate was arbitrarily chosen. The results of these analyses are post-processed in *Salome-Meca*.

At first a setting for the mesh granularity has to be identified, which delivers accurate results while keeping the calculation time low. The WF_Candidate2 study loaded only with the clamping force is used. The maximum line length of the elements of the jaws and the WP are varied, which define the element size, and the resulting magnitude of the displacement of the active surface is plotted over a path parallel to the x-axis, i.e. along the WP width. Figure 6-16 shows the results of the study. In the bottom right corner of the plot, the active surface and the path that the node values are taken from are shown.

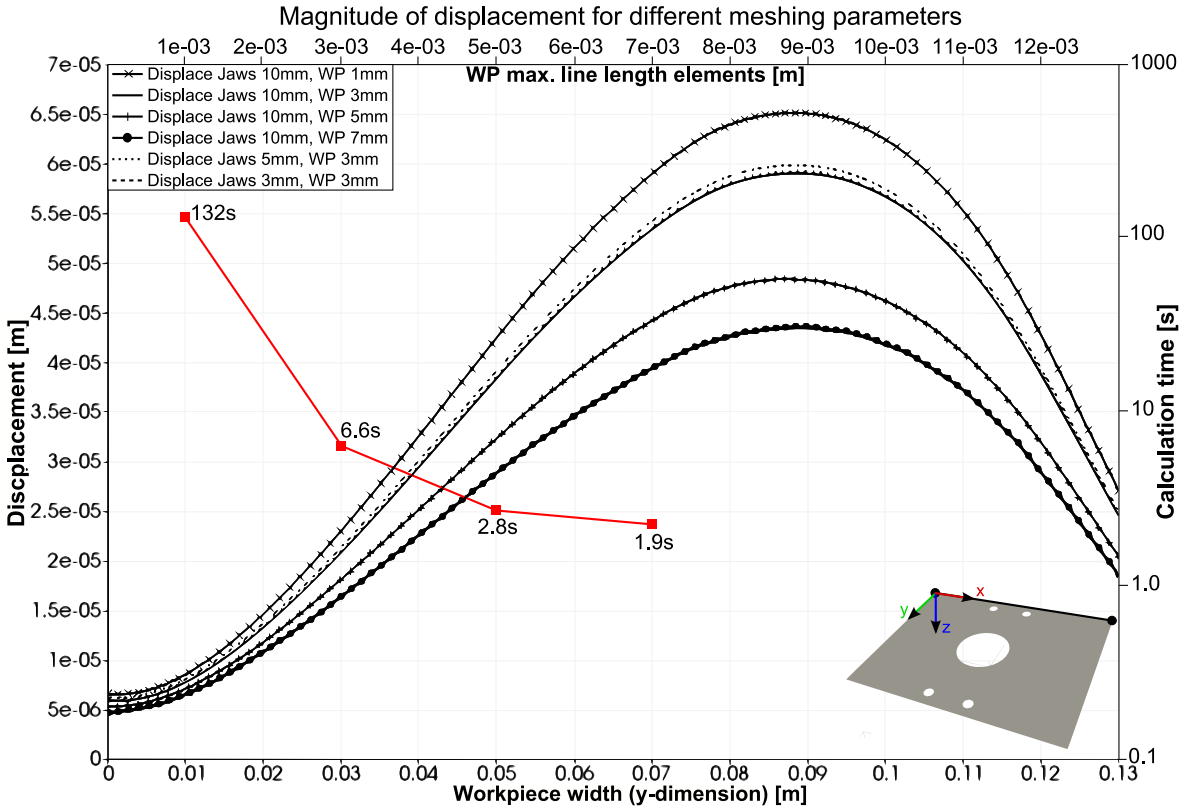



**Figure 6-16 Calculation time and displacement magnitude of the active surface of Winged_Flange_Setup1of3_Cand2 depending on the maximum mesh element line length**

It can be seen that the element size of the jaws has little effect on the displacement results. However, the maximum resulting displacement increases when the element size of the WP decreases. The square marked line, which shows the calculation time depending on the element size of the WP, depicts that a decrease of the element size drastically increases the required

time. As a result of this study, a maximum element line length of 3mm for the WP and 10mm for the jaws is chosen for the subsequent analyses, as these settings deliver adequately accurate displacement results while keeping the calculation time low (6.6s).

To determine the critical deformation direction the x-, y- and z-displacement of the active surface of study WF_Candidate2, loaded with the clamping force and the machining force of feature 1, are checked. Figure 6-17 shows the displacements plotted over two lines, one parallel to the x-axis and one parallel to the y-axis, as depicted in the bottom right corner of the figure. The gaps in the lines are caused by the central hole of the part. The graph shows that the z-displacement reaches the maximum values of all displacements along both paths. Therefore, the deflection of the WP in z-direction is the critical deformation.

To verify the correct application of the clamping and machining forces, the area of the maximum deformation is checked. Considering the bottom axis and the curves it can be seen that the absolute maximum occurs at the outermost WP edge in the y-direction (y=0mm) and, approximately, in the middle of the WP in the x-direction (x=70mm), with respect to the local WP COS. This is feasible as the center of the WP in the x-direction is at 65mm and here the maximum deflection would be expected due to buckling. Because of the different widths of the supporting surfaces of the jaws in combination with the gluing of contact surfaces, the resulting position is slightly shifted. In the y-direction, the maximum displacement occurs at the innermost edge (y=0mm) as the WP is not completely clamped along its length, such that the outermost edge of the active surface (y=100mm) is less subject to the clamping forces.

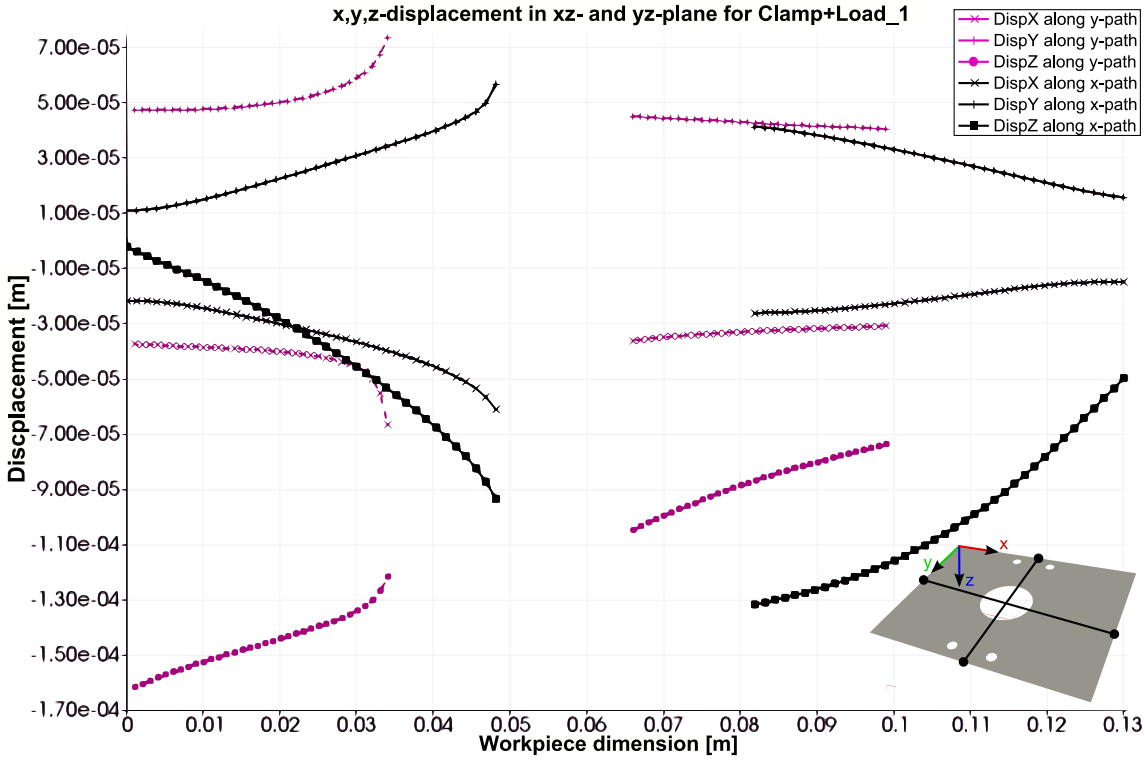



**Figure 6-17 x,y,z-displacements of the active surface of study Winged_Flange_Setup1of3 loaded with the clamping force and machining force of feature 1 plotted over two central lines (see bottom right)**

As the initial studies show qualitatively feasible results, the six evaluation studies can be carried out. In these analyses all load cases are applied independently and the resulting nodal displacements of the active surface are stored to determine the maximum displacement among all load cases. Although it was determined that the z-displacement is the most critical in the initial study, all displacement values are stored to check if this condition holds for all studies. Figure 6-18 shows an exemplary plot of the z-displacement results for study WF_Candidate2. It can be seen that the load-case Clamp+Load_1 causes the biggest displacements with an approximate value of -1.74E-04m.

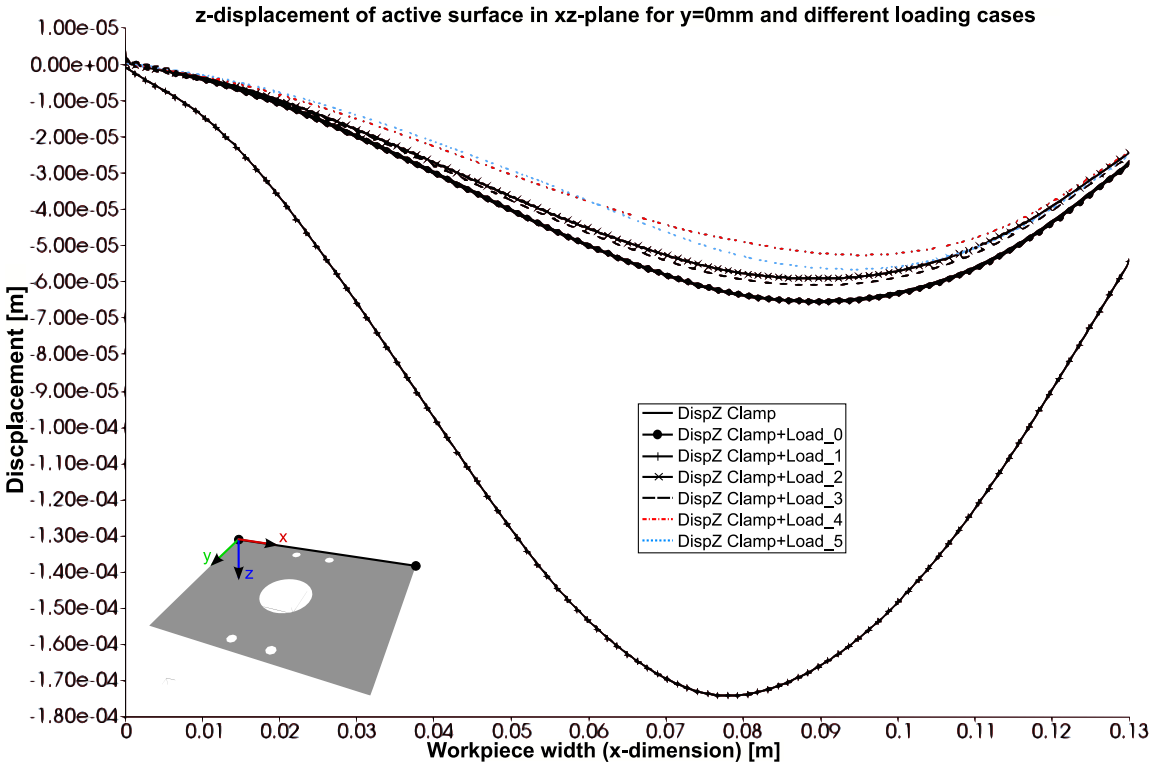



**Figure 6-18 z-displacements of the active surface of study Winged_Flange_Setup1of3 plotted over one path parallel to the x-axis at y=0mm for all different load-cases**

To determine which of the FD candidates for a specific setup causes the least WP deformation the maximum displacement results are compared. The results of the CYL setup are given in Table 6-4 and the complete table of results for all six studies is given in Appendix 10.11. For all candidates among both studies the minimum z-displacements are the biggest absolute values, which means that like in the initial study the buckling caused by the clamping force is the most dominant deformation. For the CYL study, the first candidate causes the smallest maximum displacement value (see Table 6-4). Among the three tested, this candidate would be used in production as it causes the least WP deformation and, thus, the highest WP accuracy.

**Table 6-4 Nodal displacements for the active surface of all three FD candidates of study Cyl_D60_H15 Setup1of1 resulting from the FEA given in [m]**

| Cyl_D60_H15_Setup1of1_Candidate1 | Min DX | Max DX | Min DY | Max DY | Min DZ | Max DZ |
|---|---|---|---|---|---|---|
| Clamp | -3,89E-05 | -1,03E-05 | -2,32E-05 | 1,90E-06 | **-7,41E-05** | 1,58E-06 |
| Clamp+Load_0 | -3,29E-04 | -1,05E-04 | -2,74E-04 | -8,30E-05 | **-5,67E-04** | 4,27E-05 |
| Clamp+Load_1 | -5,05E-05 | -1,15E-05 | -2,80E-05 | 4,04E-07 | **-8,96E-05** | 1,73E-06 |
| Clamp+Load_2 | -4,86E-05 | -1,16E-05 | -3,00E-05 | 9,67E-07 | **-8,79E-05** | 1,77E-06 |
| Clamp+Load_3 | -4,34E-05 | -7,50E-06 | -2,39E-05 | 1,04E-06 | **-5,04E-05** | 1,23E-06 |
| **Cyl_D60_H15_Setup1of1_Candidate5** | **Min DX** | **Max DX** | **Min DY** | **Max DY** | **Min DZ** | **Max DZ** |
| Clamp | -6,20E-05 | -2,04E-05 | -1,80E-05 | 1,56E-06 | **-1,18E-04** | 5,89E-06 |
| Clamp+Load_0 | -5,93E-04 | -1,48E-04 | -3,51E-04 | -1,29E-04 | **-8,94E-04** | 9,78E-05 |
| Clamp+Load_1 | -7,78E-05 | -2,51E-05 | -2,53E-05 | -7,13E-07 | **-1,37E-04** | 6,71E-06 |
| Clamp+Load_2 | -7,47E-05 | -2,64E-05 | -2,77E-05 | -3,07E-07 | **-1,35E-04** | 6,63E-06 |
| Clamp+Load_3 | -5,89E-05 | -2,31E-05 | -1,77E-05 | 1,27E-06 | **-8,59E-05** | 4,66E-06 |
| **Cyl_D60_H15_Setup1of1_Candidate7** | **Min DX** | **Max DX** | **Min DY** | **Max DY** | **Min DZ** | **Max DZ** |
| Clamp | -5,05E-05 | -1,46E-05 | -4,22E-05 | 1,27E-06 | **-1,00E-04** | 3,72E-06 |
| Clamp+Load_0 | -3,87E-04 | -1,03E-04 | -3,19E-04 | -1,00E-04 | **-6,86E-04** | 7,29E-05 |
| Clamp+Load_1 | -6,15E-05 | -1,87E-05 | -4,89E-05 | -8,58E-07 | **-1,16E-04** | 4,10E-06 |
| Clamp+Load_2 | -6,09E-05 | -2,04E-05 | -5,24E-05 | -4,30E-08 | **-1,14E-04** | 4,12E-06 |
| Clamp+Load_3 | -4,72E-05 | -1,43E-05 | -4,30E-05 | 4,95E-07 | **-7,66E-05** | 2,92E-06 |

## 6.2.4 Discussion of the FEA

The results show that although the FD candidates for one setup may only slightly differ by their geometric parameters, these variations can have a significant effect on the maximum displacement values. The biggest and smallest maximum displacement values of the candidates of the WF setup differ by 89% and those of the CYL setup differ by 58%. The evaluation studies further show that the deflection in z-direction is the critical WP deformation with the given fixture device and independent of the machining operations carried out. For a WP with a height much larger than the clamping width, this can be different. However, according to fixture design principles, setups of this kind are disadvantageous and should be avoided. Furthermore, all minimum and maximum displacements of the active surface are considered in the comparison of the results such that the candidate with the generally smallest maximum displacement, disregarding of the displacement direction, is selected for production.

The analyses are carried out as linear, static calculations. Not considering the frictional contact between the fixture components and the WP has been shown to slightly falsify the analysis results when compared to calculations including friction and experimental validation data (AMARAL et al. 2005; ZHENG 2005). However, comparative studies showed that the results obtained under the assumption of frictionless contacts are reasonably accurate for fixture design (LEE & HAYNES 1987; KASHYAP & DEVRIES 1999). In the light of an automated system for low-volume production and the resulting frequent calculations necessary, the slightly more accurate results to be obtained do not make up for the increase in calculation time that comes with the frictional contacts and the resulting non-linear calculations.

The calculation of the machining forces using Kienzle's formulas and the way they are applied as loads in the analysis is based on several assumptions and simplifications. It has been shown in AMARAL et al. (2005) that the machining forces should be applied dynamically at the points/lines of the actual tool-WP interaction for more accurate results and that the torque induced by the tools should be included, especially in milling. However, calculating an accurate machining load condition that works in a generally applicable and automatable way is a complex problem and out-of-scope of the presented work. Further, using simplified static loads is commonly and successfully applied in WP deformation calculations.

As the machining forces directly influence the clamping force setting in the presented approach, the simplifications can have a falsifying effect. To ensure that no WP movement occurs, the calculated clamping forces are multiplied by a safety factor in addition to the conservative calculation. This can lead to clamping forces that are too high, causing unnecessary WP deformation. The analysis results showed that the clamping force has a major influence on the WP deformation as it causes buckling. Thus, determining the minimum clamping force that still causes a restriction of the WP movement during machining, i.e. force closure of the WP, would be beneficial. Determining the minimum feasible clamping force in an automated FE analysis including friction is possible, as was shown in a study carried out in the course of this work (DIERMANN 2013). Because of the very high calculation times and the increase in complexity to automate the approach it is, however, not included in the FE process at this point.

In the presented approach, the FE analysis results are used to evaluate and rank different FD candidates. Besides for evaluation, deformation analysis is commonly used for fixture design verification, by comparing the deformation results against allowed maxima. The calculation of the allowed deformations has to be based on the tolerance values of the WP with respect to machining feature dimensions and datums. As features, datums and tolerances were initially excluded from the scope of the presented research, this verification method is currently not included in the approach. Given that the maximum allowed deformation values are known or can be calculated, it can easily be added.

Further, an empirical verification of the analysis results was not carried out as the focus of the presented work lies on the generation of a FEA model formulation and a process for the automation of the analysis. To validate the FEA modeling approach and especially the machining force calculation and application as surface or edge loads, an empirical verification would be interesting.

### Automation of the FEA

The focus of the presented FEA work is on automating the analysis process for non-predefined WP geometries and different flexible vise candidates. The used software applications allow for the automatic processing of analysis using scripts. The modules used for the geometry import and meshing can be fully automated using *Python*. The *Python* scripts for one of each case study are given in Appendix 10.12. Besides the file names and paths, the scripts for different candidates only differ in the ID numbering of the geometric entities, like faces and edges, and the amount of geometric entity groups. The ID numbers are used to define the groups of geometric elements that the boundary conditions are later applied to. Thus, for all geometric entities that a boundary condition must be applied to, such as a contact face or a machining

feature, a group needs to be defined and labeled in the script. If the ID numbers of the geometric entities of the candidates of one setup are the same, i.e. if their topology is the same, the scripts can be reused without any alteration. Else, the ID numbers of the geometric entities and the amount of groups, depending on the number of features, have to be changed.

Running the *Python* script creates a meshed geometry with groups of mesh elements. Based on this, the analysis can be run using *Eficas* script files (EDF 2014) that control the solver-module. The *Eficas* script files for the analyses are given in Appendix 10.13. In the script, the material models, the boundary conditions and the calculation type and details are defined and put into relation with the labeled groups of the meshed geometry. In the presented scripts, *Python* lists are used to generate the results for all load-cases in a loop in one study. The scripts for different studies only differ in the loads, such that the basic script can be reused for every study carried out.

To automatically run an analysis the FEM software is started in terminal mode passing the paths to the two script files of a study. This generates the results defined in the *Eficas* script, in this case the table of minimum and maximum nodal displacements of the active surface. Using this process, the presented evaluation studies can be run automatically, given that the script files for each candidate exist. Although the scripts can be reused for the most part for different candidates, the definition of the geometric groups, the force parameters and load types needs to be customized. Thus a unique script file for every candidate is required. To automate the generation of the scripts the ID numbering of the geometric entities needs to be controlled or a way to determine the ID numbers based on the geometric data is required. As the first option is not possible with the functions provided by the software, the latter seems more promising. The contact entities of the WPs and jaws are already explicitly represented in the AFD ontology so that their ID numbers can be stored in and retrieved from the ontology. To also cover the entities that the machining loads are applied to, machining features would need to be added to the ontological representation, as presented by FAN (2010). Further, the model paths, calculated force values and the material data need to be added to the scripts. All necessary information for this is stored in the ontology and can be retrieved from there. An automated script generation is, however, not covered in this thesis.

## 6.3 Conclusion

The section presented the analysis methods used for verification and evaluation of the FD candidates generated either through ontological reasoning or through computational design synthesis. This covers tool-fixture interference analysis and WP deformation analysis.

For the interference analysis an approach and corresponding implementation that builds and uses the assembly model of the WP and the fixture components as well as the volume that the cutting tools require for the machining of the setup is presented. The main tasks for this analysis are the extraction of tool movements from the NC file of the setup, the generation of the TSV based on sweeping 3D cutting tool models along the extracted tool paths, the generation of the assembly models and the test if the models interfere with each other. The functionality of the approach and the implementation was verified by successfully applying it to the case-study parts.

The FE method is used for the deformation analysis. The FEA using the presented tool set and modelling approach allows for a detection of the FD candidate that causes the least WP deformation among the interference-free candidates. It therefore enables the selection of the best candidate among the existing ones with respect to WP deformation. The analysis determines the maximum occurring displacement of the active surface of the WP for every FD candidate in a fast static, linear calculation. The FEA approach, which is able to process all jaw set and WP topologies, is evaluated using different FD candidates for two case-study setups and the results are critically reviewed for feasibility and correctness.

Both analyses work semi- but not yet fully automatically. For the interference analysis, a custom tool based on the *OCCT* framework was implemented. Within a GUI, the NC file of a setup needs to be manually selected and a text file containing the paths to the 3D models of the parts for every FD candidate assembly needs to be present. The FEA analysis can be run using *Python*-based scripts. So far, the selection of the geometric entities of the assembly model that are subject to boundary conditions, such as fixed or loaded faces, and the machining and clamping force calculations need to be manually carried out. The resulting data then needs to be added to the script templates together with the material parameters of the WP and the loading conditions, which vary depending on the machining processes used. As the FEA tool set is partially based on the *OCCT* framework and supports general *Python* functions an increase of the level of automation of the analysis seems possible.

# 7. Discussion and Future Work

The thesis presents an approach to an automated fixture design (AFD) system for a flexible vise-type fixture. The approach covers three of the four fixture design phases: fixture planning, fixture configuration design and fixture design verification. The fixture design phase setup planning is not included in this work. For fixture planning and fixture configuration design, a formal ontology with knowledge reasoning and a spatial grammar for jaw set design synthesis are used. For design verification and evaluation, tool-fixture interference and workpiece (WP) deformation analyses are carried out.

The AFD system can generate a flexible vise fixture design for a given workholding job and setup plan. The system will first attempt to reuse existing fixture components and if no feasible design can be created with these, new designs are generated. The design verification guarantees that all candidates are free of geometric interferences. Among these final candidates the best one, with respect to the least maximum WP deformation, is determined and used for the machining of the WP. All 3D models of this candidate design that are created during the process are added to the model repository and the fixture design and component data is added to the ontology for potential re-use.

In the following sections, the research contributions of the presented work, the capabilities of the developed methods and approaches for each fixture design task, their integration in the formal fixture design (FD) representation, their interoperability and also the limitations of the approach are discussed.

## 7.1 Research Contributions

The thesis presents the basis for an AFD system that integrates a formal knowledge representation with methods for fixture configuration design by reasoning and design synthesis, verification and evaluation. Further, it links FD to the physical reconfiguration of a flexible fixture device on the hardware level. Known shortcomings of previous computer-aided and automated fixture design research were considered in the selection and implementation of the methods and approaches used. In the following, the key research contributions of this thesis are summarized shortly before they are presented in more detail.

- The developed formal ontology for FD knowledge representation is the first FD ontology that covers representation of vise-type fixtures and includes fixture functions and locating principles. Further, the developed fixture design representation scheme allows for the determination of feasible fixture designs for new WPs through ontological reasoning. The reasoning process is based on matching the fixture capabilities, i.e. the capability to locate, support or clamp geometric WP entities of a certain kind and shape, of the fixture components of each automatically generated fixture instance to the geometric entities of a WP. This is a new approach to fixture configuration design and a first proof-of-concept of the use of ontological reasoning for an engineering design synthesis and physical embodiment reasoning task known to the author. In comparison to other FD ontologies

the upper ontology structure developed is independent of the fixture type and, thus, extendable and reusable in other scenarios or systems.

- The developed spatial grammar allows for requirement-driven computational synthesis of production-ready fixture component designs. The use of computational design synthesis for the generation of problem-specific fixture components has not been presented before. The use of design space constraining obstacles, which include an interference check in the design configuration phase, also has not been presented previously in the field of fixture design. Further, spatial grammars have rarely been applied to real, highly constrained engineering design problems and the generation of detailed designs as done in the presented work. The results are, thus, a proof of the applicability of spatial grammars and of the used spatial grammar interpreter *Spapper* for a detailed design task.
- The approach and application for tool-interference analysis uses the known tool swept volume method but, compared to similar approaches, works automatically, directly derives all required data for the sweeping operations from an external NC file of a WP and can process not just cylindrical but any rotationally symmetric tool shapes, including tool holders and other peripherals. The generated swept volumes are not only used for interference analysis but also as obstacles in the design synthesis approach. The approach and the application are independent of other FD approaches used and are not limited to tool-fixture collision detection. An extension or reuse in other scenarios, also of single parts of the application, is possible. For example, the G-code interpreter can be adapted to extract data required for a machining force calculation from NC files.
- As CAFD and AFD research primarily focusses on modular fixtures, research into the automation of the design and configuration of a vise-type fixture is an extension to the state-of-the-art.

## 7.2 Final Discussion of Approach

### Fixture Design Ontology

The developed ontology serves as an integrated information model and knowledge base for the fixture design task and all involved applications. Through the use of this formal AFD ontology fixture design information is made available in all steps of the FD process and the overall design-to-fabrication process. As discussed by PEHLIVAN & SUMMERS (2008), the generation of such an integrated FD representation, which allows for information transfer between the different phases of fixture design, is a key aspect in fixture design research that is not yet solved. The presented ontology extends previously existing FD ontologies, like those presented by (AMERI & SUMMERS 2008) and (FAN 2010), to representation of vise-type fixtures. Further, the developed representation is not just based on geometric and component information but represents locating principles via the fixture functions *locates*, *supports*, *clamps* and *holds*.

Besides serving as an information and knowledge model, the ontology allows for reasoning about the feasibility of fixture candidates for given workholding jobs. The fixture candidates are generated automatically based on the fixture type definitions and the fixture component data modeled in the ontology. By using anonymous class descriptions and SWRL rules the fixture

functions of each fixture component and each fixture instance are added to the ontology by an ontology reasoner which can then, driven by manual queries, be used to semi-automatically determine a geometrically feasible fixture configuration design for a WP. For this, the geometric WP data needs to be part of the manual query or added to the ontology. Although currently focused on the representation of and reasoning about flexible vise-type fixtures, the general extendibility of ontologies, the generality of the representation scheme and the variety of fixture types and components already loosely defined in the upper fixture ontology allow for an extension and application of the ontology to other fixture types and other FD scenarios.

### Computational Synthesis of Fixture Components

In case no feasible fixture configurations can be determined based on the ontological reasoning, new, problem-specific designs of jaw sets can be generated automatically using a spatial grammar and a given WP model that is oriented and labeled according to the setup plan. The generation includes several geometrical and functional design requirements that are embedded in the vocabulary and rules of the grammar. This drastically reduces the need for subsequent analysis of the designs. Most notably, by using solid obstacles, which represent the volume that the cutting tools require for the machining of a setup, the synthesis of interference-free designs can be guaranteed. The list of requirements is, however, not exhaustive. Aspects of the machining envelope are, for example, not covered. Although cumbersome, an extension of the grammar to include further design requirements is possible by changing the respective rule definitions. This was proven during the grammar development, as several requirements were iteratively embedded.

### Tool-Fixture Interference Analysis

For fixture design verification a tool swept volume (TSV) approach is used. This approach generates solid models of the volume that the cutting tools require during their travel along the cutting paths. The volumes generated this way are used in two ways. First, they can be used in an interference analysis to detect collisions of the tools with the fixture candidates generated by ontological reasoning. Secondly, they can be used as obstacle volume in the design synthesis which guarantees for the generation of interference-free jaw set designs. Collision detection methods provided by commercial CAD/CAM solutions do not generate an exportable solid model that could be used for this task.

The TSV approach has already been successfully applied in the area of modular fixture design, as presented in (SENTHIL KUMAR et al. 2000). In the work at hand, the approach is extended to allow for the automatic generation of the TSV models and automatic interference analysis. For this, the used tool and tool path data for the machining of a setup are derived directly from the NC code of the part. Together with a set of 3D tool models, that can include further objects such as tool holders, this data is used to generate the TSV models. These TSV models can directly be used as obstacles in the design synthesis process or in the automatically generated assembly models of the fixture candidates that further include the fixture components and the WP. Based on the assembly models the analysis of interferences between the TSV and the fixture components is carried out.

The approach is not limited to the presented fixture device or even to the area of fixture design but can be generally applied to NC machine tool collision detection. The NC code data extraction can be extended to cover more tool movement commands or other commands, such as the feed-rate and spindle-speed commands. Further, the approach is not bound to predefined tool models. Other tool models can simply be added to the tool model library and used, given that they are modeled in the way presented in Section 6.1.3.

### Workpiece Deformation Analysis

WP deformation caused by the clamping and machining forces is a critical issue for the achievable WP accuracy (AMARAL et al. 2005). FE analysis, as a standard method to calculate such deformations, is used in the presented approach. Based on the results of the analysis all existing fixture configuration candidates are evaluated and ranked in order to pick the candidate that causes the smallest maximum WP deformation.

The evaluation based on the case study parts showed that automation of the analysis process with the defined model formulation and approach is generally possible, although two aspects in the generation of the scripts that drive the analysis still have to be carried out manually so far. These are the numbering and grouping of the contact entities and the definition of parameter values. Further, the calculation of the machining loads is not yet automated. All necessary data can be derived from the NC file in combination with the tool model and material data stored in the ontology. The NC-file interpreter presented in Section 6.1.2 already extracts the required tool path data. It could be extended to extract feed-rates and spindle speeds required for the force calculation. An ontological representation of the features could then be used to store the force values and make them available for the FEA script generation. Additionally, a generalization of the force application would help to simplify the problem.

### Application Interoperability

A known shortcoming of computer-aided FD research that is still not satisfactorily solved is the lack of methods that seamlessly use the information provided by CAD models and are interfaced with CAM and manufacturing resources (CECIL 2001; PEHLIVAN & SUMMERS 2008). The presented approach addresses this lack as all involved methods use the information provided by the CAD representation of the WP and the fixture components and focus on the ability to interact with machining planning, fixture hardware reconfiguration and manufacturing system control. All applications that include geometric models are based on the same open source geometry kernel *Open Cascade*. A common file type for 3D models, the *.step protocol, is used when possible. Additional data, such as the setup function of WP surfaces, needs to be explicitly added to the CAD representations for some methods. The storing of this data, added in the form of 3D labels, requires the use of an additional data type, the native *FreeCAD* format.

All input data for the spatial grammar, such as the path to the WP and the obstacle model, is stored in the AFD ontology. Similarly, the input data required for the assembly model generation and interference analysis can be retrieved from the AFD ontology and model repository. Newly generated models, such as synthesized jaw designs or assembly models can be added to the AFD system model repository and represented in the AFD ontology. The results

of the interference analysis can be stored as Boolean true or false property in the ontology. Most of the input data required for the FE analysis, such as the assembly models and NC files of the WPs, is stored in the AFD system and linked in the AFD ontology. Also the material data for both the jaws and the WP can be retrieved from the AFD ontology. The geometric entities of the jaws and WPs are explicitly available and marked with their setup function. This information could be used to drive the generation of the scripts required for automating the FEA.

### Link to Fixture Reconfiguration

The actual data transfer between the software part of the AFD system and the fixture hardware reconfiguration has not been implemented but is carried out manually. All required data is generated by the applications and can be stored in the AFD ontology in a machine interpretable way. The only exception is the generation of NC code for new jaw designs created by the spatial grammar. Using the generated 3D models and the existing raw jaw model a NC code can be created manually with a CAD/CAM application or automatically using a machining planning method, such as the spatial grammar based approach presented by ERTELT (2012). Automated machining planning is so far not included in the approach and the NC codes are created manually. The path to the NC code can then be added to the ontological representation of the setup. Using only the data present in the ontology and manually providing it to a PLC-based control system, a reconfiguration of the flexible fixture device and machining of a new set of vise jaws has successfully been carried out, as presented in a study done in the course of this work (FANG 2012). By developing, e.g., an agent-based FMS control system that allows for direct communication of the required information from the ontology to the FMS hardware, fully automatic reconfiguration and adaption can be achieved.

### Vise-Type Fixtures vs. Modular Fixtures

When looking at the state-of-the-art in computer-aided or automated flexible FD research, mainly modular fixtures are discussed, as they are the most commonly used type of flexible fixtures in industry. In comparison to modular fixtures vise-type fixtures can only fix a limited range of part shapes, as e.g. parts with complex freeform surfaces cannot be held securely with vises. However, they still are frequently used not only in manual shop-floors but also in manufacturing systems of all kind because they allow for a fast setup of WPs, offer a better cost-benefit ratio than modular fixtures and provide a sufficient flexibility for many fixture tasks. Using a vise-type fixture like the flexible vise, especially as part of an AFD system, can, therefore, not replace modular fixtures completely but can be an enhancement for manufacturing systems in low-volume production scenarios that reduces setup costs and increases the degree of flexibility and automation of the overall system.

## 7.3 Limitations

The presented AFD system has limitations that result from the flexible vise, the approaches used for the FD tasks and the implementations of the approaches. The fixture device imposes several limitations on the processable workholding jobs. As shown, the vertical locating of cylindrical WP is possible with horizontal V-blocks. Following this locating principle no exact

rotational positioning of the WP is possible without using additional contact entities, like a plane surface. The use of such additional features is not considered in the approach or supported by the vise. Nevertheless, through appropriate setup planning many situations where accurate rotational positioning is required can be avoided.

The horizontal clamping of cylindrical WPs is not possible as this would require vertical V-block locating. Jaws of that shape cannot be machined with the fixture device and manufacturing system given and are, thus, not considered in the approach. Further, the WP shape restrictions presented in Section 3.8 cannot be overcome with the jaw shapes used. An extension to cover other jaw shapes, which resemble other locating principles such as inside hole locating, or allow for pull-type clamping in comparison to the used push-type clamping, would require a rework of all approaches and applications.

Considering the approaches and applications, aspects of automation and achievable design quality are the most critical. A point that requires manual work and is error-prone is the instantiation of data in the AFD ontology. The data required for the applications needs to be derived from the WP and fixture component models and the NC codes of the WPs and added to the ontological representation. The integration of a geometric kernel with the ontology application programming interface could be used to automate this instantiation process. Further, the ontology needs to be extended to work as integrated information model for all applications. Most important aspects for the presented system are already covered but aspects like the displacement boundary conditions required as input to the FEA or storing the FEA results have not yet been included. It was shown in previous research that a representation of this data in an ontology is possible (FAN 2010). A related extension is, therefore, considered feasible. Furthermore, the communication of the applications with the ontology is not (yet) automated but based on manual interaction. Thus, the interoperability of the applications and the ability to automate the FD system is not formally verified so far. Many examples of automated application communication based on ontologies, e.g. through software agents, have been presented in literature. An according implementation is therefore, again, considered feasible.

The fixture configuration design based on reasoning currently only works for the flexible vise or, with minimal adaptions, for single-acting vises in general. Covering other fixture types is theoretically possible but requires additional classes and SWRL rules that could affect the existing reasoning logic. Thus, expanding the ontology accordingly may require a rework of the existing fixture representation scheme.

The developed spatial grammar is specific to the generation of jaws for the flexible vise. Adapting the grammar to the synthesis of jaws with a slightly different geometry, e.g. other maximum dimensions, is possible. A broader reuse for other fixture component or fixture configuration design tasks, however, is not possible. This is a common limitation of spatial grammars that are generally built for a specific design task.

The tool-fixture interference application is currently limited to straight and circular 3D movements and 2.5D machining with a vertical tool approach direction. An extension to include sweeps along more complicated trajectory paths, such as curved lines, and for multi-axis machining is theoretically possible, as the approach is not restricted in this way. However, such sweeping operations are known to be complicated in creation, e.g. because of self-intersections of the resulting volumes when small curvatures occur in the paths. Thus, their inclusion in the

application may be difficult. Further, representing tool rotations, as required for multi-axis machining, will require substantial extensions of the application. Also, no solution for an automated trimming of the generated TSV could be determined so far. Without trimming, the resulting TSV files cause a slow and unreliable performance of the interference check and the spatial grammar. Splitting the TSV into several smaller models, as currently done in the AFD system to make the approach work, can only be seen as a workaround. A solution for trimming the overall TSV files during or after their generation is still required. The limited trimming capabilities of the used geometric kernel are known to the developers and actively being worked on, such that a future release may resolve the issue. Additionally, further trimming approaches as presented by BLACKMORE et al. (1999) could be tested.

Considering the FE analysis, mainly the degree of automation achievable for this application and the quality of the analysis results are currently limited. For automation, the generation of the script files that drive the analysis needs to be automated. This requires communication with the ontology for data retrieval and the flexible generation of groups of geometric entities that the boundary conditions can be applied to as well as the calculation of the machining and clamping forces. These aspects have already been discussed in Section 6.2.4. Considering the broader applicability, the presented scripts are limited to the specific assembly model structure used, i.e. a jaw set and a WP. Both the approach and the application are not restricted to this but an extension to other fixture types or assembly model structures requires new scripts. Another limitation of the presented FEA model is the lack of an experimental validation of the analysis results. To ensure that the model represents the reality and to check the quality of the results, such a validation is required. Based on the results, a refinement of the model and the boundary conditions may become necessary.

Finally, aspects of locating accuracy of the fixture are not included so far as WP datum information or tolerances are not considered. This limits the usability of the current system in industrial scenarios, where locating accuracy is a key concern, and makes deformation verification and tolerance analysis impossible. However, all of the developed models and approaches can theoretically be extended to respect datum and tolerance information.

## 7.4 Recommendations for Future Work

Besides addressing the mentioned limitations further extensions and studies are required to fully automate workpiece fixturing as part of the design-to-fabrication process and to test the flexibility and applicability of the presented AFD system.

To complete the fixture design process an automated way of setup planning that can interact with the other applications and the ontology is required. In literature, several different approaches have been presented. Most approaches are not fully automatic but require some user interaction, either to guide the setup planning or to select certain setups among the possible ones. To allow for easy integration with the existing system, the setup planning should include the premises discussed in Section 3.8 and the process depicted in Appendix 10.5. Additionally to the interference analysis, design verification methods for tolerance and deformation analysis should be included to allow for a holistic verification of FD candidates, as illustrated in Section 2.2.3. To make use of the dynamic clamping force control that the flexible vise offers, advanced FEA formulations could be used to determine an optimal clamping force for every

state of the machining process and, thus, minimize WP deformation. The sole use of *.step files in an application protocol that supports annotations would eliminate the need for storing (partially redundant) 3D model data, as done in the current version. This can be realized as soon as one of these protocols is supported by the geometric kernel used for all applications (*Open Cascade Technology*). However, this would also require the rework of the spatial grammar interpreter to use this annotation type for labeling. A final step in the AFD system development would be to create an integrated software environment that combines all applications in one user interface, in order to facilitate the use and increase the acceptance among the practitioners in industry that should later use the system.

Finally, the AFD system should be integrated with the WP selection and the machining planning approach presented in (SHEA et al. 2010) to create a combined computational design-to-fabrication system for the automatic processing of non-predefined parts. Further integrating this combined system with the local and global planning of a manufacturing system would result in a large step forward towards an autonomous and highly flexible manufacturing system.

Apart from the extension of the AFD system its applicability should be evaluated. An industrial case study in a low-volume production scenario using the flexible vise and AFD system can be run to test the flexibility, degree-of-automation and robustness of the system and to carry out an economical study on the costs for fixture design and fixture adaption/reconfiguration in comparison to other flexible fixtures. In general, a quantitative comparison of the use of flexible fixture versus dedicated fixtures and the related costs in today's industry would be of great importance to both determine focus areas for FD research and to show possible benefits of (C)AFD systems and drive their acceptance in industry.

Although the presented approaches are applied to and verified with a specific use case, the flexible vise, extending and applying the approaches and applications to other design automation tasks in the area of manufacturing automation may be interesting. These tasks include, but are not limited to, automated transport pallet design, robot gripper design and defining a formal representation for parts that explicitly includes setup and fixturing information.

# 8. Conclusions

Automated fixture design and the automatic adaption or reconfiguration of flexible fixture devices to different workpiece shapes is an essential aspect to increase the flexibility and degree of automation of a manufacturing system. In the light of low-volume production with frequent change-overs, the impact of fixture design and fixturing becomes even more important.

This thesis presents a new automatic fixture design system that covers the fixture design tasks of fixture planning, fixture configuration design and design verification and combines them with a formal knowledge representation and the physical reconfiguration and adaption of a flexible vise-type fixture device. The fixture design tasks are addressed with either established methods with new extensions or foci or with novel methods that have not been applied to these tasks before. An ontology is used not only for formal knowledge representation and as an integrated information model but also for fixture configuration reasoning, allowing for the generation of fixture candidates for new workholding jobs from the existing fixture components. This is based on a representation of the geometry of the workpieces and fixture components that covers fixture functions of the geometric entities in combination with a representation of locating principles in the form of rules. Further, a spatial grammar allows for the requirement-driven synthesis of new, problem specific fixture components and fixture candidates. A tool-fixture interference analysis that is driven by the NC codes of the workpieces is used for design verification. Candidates passing the verification can then be ranked using a workpiece deformation analysis to determine the best fixture design among the created alternatives. The presented results illustrate the feasibility of the proposed methods and developed applications.

In its current state, the system allows for the generation of fixture designs in a semi-automated way and for the fully automatic reconfiguration and adaption of the fixture device on the hardware level by providing the generated design data to the hardware control systems. The system, therefore, allows for an increase of the degree of flexibility and automation of the fixture design and fixturing process and, by formally encoding the fixture design knowledge, reduces the need for user expertise to design and configure a fixture. The ability to fully automate the system has been discussed and is subject to future work.

# 9. References

AGARWAL et al. 2000

Agarwal, M.; Cagan, J.; Stiny, G.: A micro language: generating MEMS resonators by using a coupled form-function shape grammar. Environment and Planning B: Planning and Design 27 (2000) p. 615-626.

AMARAL et al. 2005

Amaral, N.; Rencis, J. J.; Rong, Y.: Development of a finite element analysis tool for fixture design integrity verification and optimisation. The International Journal of Advanced Manufacturing Technology 25 (2005) 5, p. 409-419.

AMERI & SUMMERS 2008

Ameri, F.; Summers, J. D.: An Ontology for Representation of Fixture Design Knowledge. Computer Aided Design and Applications 5 (2008) 5, p. 601-611.

ANTONSSON & CAGAN 2001

Antonsson, E. K.; Cagan, J. (Eds.): Formal engineering design synthesis. Cambridge: Cambridge University Press 2001. ISBN: 0-521-79247-9.

ASADA 1985

Asada, H.: Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures. Robotics and Automation, IEEE Journal of 1 (1985) 2, p. 86-94.

ASANTE 2008

Asante, J.: A combined contact elasticity and finite element-based model for contact load and pressure distribution calculation in a frictional workpiece-fixture system. The International Journal of Advanced Manufacturing Technology 39 (2008) 5-6, p. 578-588.

BAI & RONG 1998

Bai, Y.; Rong, Y.: Modular fixture element modeling and assembly relationship analysis for automated fixture configuration design. Engineering Design and Automation 4 (1998) p. 147-162.

BANNAT et al. 2011

Bannat, A.; Bautze, T.; Beetz, M.; Blume, J.; Diepold, K.; Ertelt, C.; Geiger, F.; Gmeiner, T.; Gyger, T.; Knoll, A.; Lau, C.; Lenz, C.; Ostgathe, M.; Reinhart, G.; Roesel, W.; Ruehr, T.; Schuboe, A.; Shea, K.; Stork genannt Wersborg, I.; Stork, S.; Tekouo, W.; Wallhoff, F.; Wiesbeck, M.; Zaeh, M. F.: Artificial Cognition in Production Systems. Automation Science and Engineering, IEEE Transactions on 8 (2011) 1, p. 148-174.

BAUSCH & YOUCEF-TOUMI 1990

Bausch, J.; Youcef-Toumi, K.: Kinematic methods for automated fixture reconfiguration planning. In: IEEE International Conference on Robotics and Automation, Cincinnati, OH , USA 13-18 May 1990. p. 1396-1401. ISBN: 0-8186-9061-5

BI & ZHANG 2001

Bi, Z.; Zhang, W.: Flexible fixture design and automation: review, issues and future directions. International Journal of Production Research 39 (2001) 13, p. 2867-2894.

BLACKMORE et al. 1999

Blackmore, D.; Samulyak, R.; Leu, M. C.: Trimming swept volumes. Computer-Aided Design 31 (1999) 3, p. 215-223.

BORST 1997

Borst, W. N.: Construction of engineering ontologies for knowledge sharing and reuse. Doctoral thesis, Center for Telematics and Information Technology, Universiteit Twente, Enschede, Netherlands (1997). <http://doc.utwente.nl/17864/1/t0000004.pdf>

BOYLE et al. 2006

Boyle, I.; Rong, K.; Brown, D.: CAFixD: A case-based reasoning fixture design method. Framework and indexing mechanisms. Journal of Computing and Information Science in Engineering 6 (2006) 1, p. 40.

BOYLE et al. 2011

Boyle, I.; Rong, Y.; Brown, D. C.: A review and analysis of current computer-aided fixture design approaches. Robotics and Computer-Integrated Manufacturing 27 (2011) 1, p. 1-12.

BRAY et al. 2008

Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.; Yergeau, F.: Extensible markup language (XML) 1.0 (Fifth Edition). W3C Recommendation <http://www.w3.org/TR/REC-xml/#sec-origin-goals> (2008)

BROST & GOLDBERG 1994

Brost, R.; Goldberg, K.: A complete algorithm for synthesizing modular fixtures for polygonal parts. In: IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 1994. p. 535-542.

BROST & PETERS 1998

Brost, R.; Peters, R.: Automatic design of 3-d fixtures and assembly pallets. The International Journal of Robotics Research 17 (1998) 12, p. 1243.

BROWN et al. 1994

Brown, K.; McMahon, C. A.; Williams, J.: A formal language for the design of manufacturable objects. In: Proceedings of the IFIP TC5/WG5. 2 Workshop on Formal Design Methods for CAD, 1994. Elsevier Science Inc. p. 135-155. ISBN: 0444819703.

CAGAN 2001

Cagan, J.: Engineering shape grammars: where have we been and where are we going? In: Antonsson, E. K. et al. (Eds.): Formal Engineering Design Synthesis. Cambribge, UK: Cambridge University Press 2001, p. 65-92. ISBN: 0-521-79247-9.

CECIL 2001

Cecil, J.: Computer-Aided Fixture Design – A Review and Future Trends. The International Journal of Advanced Manufacturing Technology 18 (2001) 11, p. 790-793.

CENTER FOR EDESIGN 2005

Center for eDesign: e-Design Framework 2.0 - Materials 2.0 Ontology <http://edesign.ecs.umass.edu/ontologies/Framework2.0/Materials2.0.owl> 2014 February 12.

CHAKRABARTI et al. 2011

Chakrabarti, A.; Shea, K.; Stone, R.; Cagan, J.; Campbell, M.; Hernandez, N. V.; Wood, K. L.: Computer-Based Design Synthesis Research: An Overview. Journal of Computing and Information Science in Engineering 11 (2011) p. 10.

CHANG et al. 1998

Chang, T.-C.; Wysk, R. A.; Wang, H.-P.: Tooling and Fixturing. In: Computer-aided manufacturing. 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall 1998, ISBN: 013754524.

CHASE 2002

Chase, S. C.: A model for user interaction in grammar-based design systems. Automation in construction 11 (2002) 2, p. 161-172.

CHAU et al. 2004

Chau, H. H.; Chen, X. J.; McKay, A.; dePennington, A.: Evaluation of a 3D Shape Grammar Implementation, Design Computing and Cognition. Cambridge, USA, 2004.

CHEN et al. 2008

Chen, W.; Ni, L.; Xue, J.: Deformation control through fixture layout design and clamping force optimization. The International Journal of Advanced Manufacturing Technology 38 (2008) 9-10, p. 860-867.

D. MENGEMANN SPANNTECHNIK GMBH 2014

D. Mengemann Spanntechnik GmbH: Modulare Vorrichtungssysteme <http://www.mengemann-spanntechnik.de/tl_files/mengemann/Inhaltsbilder/03-Modulare_Vorrichtungssysteme/03-09.JPG> 2014 July 16.

DAI et al. 1997

Dai, J.; Nee, A.; Fuh, J.; Kumar, A.: An approach to automating modular fixture design and assembly. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 211 (1997) 7, p. 509-521.

DE METER 1998

De Meter, E.: Fast support layout optimization. International Journal of Machine Tools and Manufacture 38 (1998) 10-11, p. 1221-1239.

DEGNER et al. 2009

Degner, W.; Lutze, H.; Smejkal, E.: Spanende Formung: Theorie, Berechnung, Richtwerte. München: Hanser Verlag 2009. ISBN: 978-3-446-41713-7.

DIERMANN 2013

Diermann, V.: Automatisierte FE-Analyse zur Verifikation von Spannvorrichtungen. Master thesis, Institue of Product Development, Faculty of Mechanical Engineering, Technische Universität München, München, Germany (2013).

DREXEL UNIVERSITY 2011

Drexel University: National Design Repository <http://edge.cs.drexel.edu/repository/> 2011 Sept. 13.

EDF 2014

EDF: Code_Aster Documentation <http://www.code-aster.org/V2/doc/default/en/?lang=en> 2014 June 15.

EDF R&D 2011

EDF R&D: SALOME 6 - The Open Source Integration Platform for Numerical Simulation. <http://files.salome-platform.org/Salome/Common/SALOME6_brochure.pdf> (2011) 2014 April 28.

ELMARAGHY 2005

ElMaraghy, H. A.: Flexible and Reconfigurable Manufacturing Systems Paradigms. International Journal of Flexible Manufacturing Systems 17 (2005) 4, p. 261-276.

ERTELT 2012

Ertelt, C.: Generative Design-to-Fabrication Automation using Spatial Grammars and Heuristic Search. Doctoral thesis, Institue of Product Development, Faculty of Mechanical Engineering, Technische Universität München, München, Germany (2012). <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20120720-1097191-0-9>

ERTELT et al. 2009

Ertelt, C.; Gmeiner, T.; Shea, K.: A Flexible Fixture and Reconfiguration Process for the Cognitive Machine Shop. In: Proceedings of the 3rd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2009), Munich, Germany, 5-7 October 2009. p. 112-120.

ERTELT & SHEA 2009

Ertelt, C.; Shea, K.: An application of shape grammars to planning for CNC machining. In: ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2009. American Society of Mechanical Engineers p. 651-660.

ERTELT & SHEA 2011

Ertelt, C.; Shea, K.: Automated Machining Planning Using Shape Grammars and Heuristic Search. Journal of Computing and Information Science in Engineering (in work) (2011)

ETSCHEIDT 1997

Etscheidt, K.: Automatisierte Montage modularer Spannvorrichtungen mit Industrierobotern. Aachen: Shaker 1997. ISBN: 3-8265-2942. (Berichte aus der Produktionstechnik 20/97). (Zugl. Aachen: RWTH, Diss.)

FAN 2010

Fan, L. Q.: Collaborative fixture design and analysis system with robustness for machining parts. Doctoral thesis, Departement of Mechanical Engineering, National University of Singapore, Singapore (2010).

FAN et al. 2010

Fan, L. Q.; Jagdish, B. N.; Senthil Kumar, A.; Anbuselvan, S.; Shung-Hwee, B.: Collaborative Fixture Design and Analysis Using Service Oriented Architecture. Automation Science and Engineering, IEEE Transactions on 7 (2010) 3, p. 617-629.

FANG 2012

Fang, Y.: Development and Validation of a PLC-Based Control System for an Autonomously Reconfigurable Fixture Device. Thesis, Institute of Product Development, Faculty of Mechanical Engineering, Technische Universität München, München, Germany (2012).

FISCHER et al. 2011

Fischer, U.; Heinzler, M.; Näher, F.; Paetzold, H.; Gomeringer, R.; Kilgus, R.; Oesterle, S.; Stephan, A.: Tabellenbuch Metall. Europa-Lehrmittel 2011. ISBN: 3808516755.

FLOHR 2014

Flohr, H.: 3D Interference Analysis using Open Cascade. Interdisciplinary Project Thesis, Institute of Product Development, Faculty of Mechnical Engineering, Technische Universität München, München, Germany (2014).

FREECADWEB.ORG 2014

FreeCADWeb.org: User Documentation - Fcstd file format <http://www.freecadweb.org/wiki/index.php?title=Fcstd_file_format> 2014 March 17.

GANDHI & THOMPSON 1986

Gandhi, M.; Thompson, B.: Automated design of modular fixtures for flexible manufacturing systems. Journal of Manufacturing Systems 5 (1986) 4, p. 243-252.

GIPS 1999

Gips, J.: Computer implementation of shape grammars. In: NSF/MIT Workshop on Shape Computation, 1999.

GIUSTI et al. 1991

Giusti, F.; Santochi, M.; Dini, G.: Robotized assembly of modular fixtures. CIRP Annals-Manufacturing Technology 40 (1991) 1, p. 17-20.

GMEINER 2008

Gmeiner, T.: Systematische Entwicklung einer modularen Spannvorrichtung für Werkzeugmaschinen. Diploma thesis, Institue of Product Development, Faculty of Mechanical Engineering, Technische Universität München, München (2008).

GMEINER 2014

Gmeiner, T.: Fixture Design Ontologies <https://ontohub.org/repositories/fixture-design-ontologies> 2014 September 05.

GMEINER & SHEA 2011

Gmeiner, T.; Shea, K.: Development of an Ontology for the Reconfiguration of a Vise-Type Fixture Device. In: ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Washington, D.C., USA, 28-31 August 2011. p. 251-261. ISBN: 978-0-7918-5482-2.

GMEINER & SHEA 2013a

Gmeiner, T.; Shea, K.: An Ontology for the Autonomous Reconfiguration of a Flexible Fixture Device. Journal of Computing and Information Science in Engineering 13 (2013a) 2, p. 11.

GMEINER & SHEA 2013b

Gmeiner, T.; Shea, K.: A Spatial Grammar for the Computational Design Synthesis of Vise Jaws. In: ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE 2013), Portland, Oregon, USA, 4-7 August 2013b. American Society of Mechanical Engineers

GRUBER 1995

Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human Computer Studies 43 (1995) 5, p. 907-928.

HARGROVE & KUSIAK 1994

Hargrove, S. K.; Kusiak, A.: Computer-aided fixture design: a review. International Journal of Production Research 32 (1994) 4, p. 733-753.

HITZLER et al. 2012

Hitzler, P.; Krötzsch, M.; Parsia, B.; Patel-Schneider, P. F.; Rudolph, S.: OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/> (2012)

HITZLER et al. 2008

Hitzler, P.; Krötzsch, M.; Rudolph, S.; Sure, Y.: Semantic Web: Grundlagen. Berlin: Springer 2008. ISBN: 3540339930.

HO et al. 2001

Ho, S.; Sarma, S.; Adachi, Y.: Real-time interference analysis between a tool and an environment. Computer-Aided Design 33 (2001) 13, p. 935-947.

HOFFMAN 2004

Hoffman, E. G.: Jig and fixture design. 5. ed. Clifton Park: Thomson 2004.

HOISL 2012

Hoisl, F.: Visual, Interactive 3D Spatial Grammars in CAD for Computational Design Synthesis. Doctoral thesis, Institue of Product Development, Faculty of Mechanical Engineering, Technische Universität München, München, Germany (2012). <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20120710-1096886-1-1>

HOISL 2014

Hoisl, F.: Spapper File Repository <http://sourceforge.net/projects/spapper/files> 2014 September 01.

HOISL & SHEA 2011

Hoisl, F.; Shea, K.: An interactive, visual approach to developing and applying parametric three-dimensional spatial grammars. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 25 (2011) 4, p. 333-356.

HOISL & SHEA 2013

Hoisl, F.; Shea, K.: Three-dimensional labels: A unified approach to labels for a general spatial grammar interpreter. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 27 (2013) 04, p. 359-375.

HORRIDGE & PATEL-SCHNEIDER 2009

Horridge, M.; Patel-Schneider, P. F.: OWL 2 Web Ontology Language: Manchester Syntax. W3C Working Group Note <http://www.w3.org/TR/2009/NOTE-owl2-manchester-syntax-20091027/> (2009)

HORROCKS et al. 2004

Horrocks, I.; Patel-Schneider, P. F.; Boled, H.; Tabet, S.; Grosof, B.; Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML <http://www.w3.org/Submission/SWRL/> 2014 28 August.

HOU & TRAPPEY 2001

Hou, J.-L.; Trappey, A. J. C.: Computer-aided fixture design system for comprehensive modular fixtures. International Journal of Production Research 39 (2001) 16, p. 3703-3725.

HU & RONG 2000

Hu, W.; Rong, Y.: A Fast Interference Checking Algorithm for Automated Fixture Design Verification. The International Journal of Advanced Manufacturing Technology 16 (2000) 8, p. 571-581.

HUNTER ALARCÓN et al. 2010

Hunter Alarcón, R.; Ríos Chueco, J.; Pérez García, J.; Vizán Idoipe, A.: Fixture knowledge model development and implementation based on a functional design approach. Robotics and Computer-Integrated Manufacturing 26 (2010) 1, p. 56-66.

HUNTER et al. 2006

Hunter, R.; Rios, J.; Perez, J.; Vizan, A.: A functional approach for the formalization of the fixture design process. International Journal of Machine Tools and Manufacture 46 (2006) 6, p. 683-697.

HUNTER et al. 2005

Hunter, R.; Vizán, A.; Pérez, J.; Ríos, J.: Knowledge model as an integral way to reuse the knowledge for fixture design process. Journal of Materials Processing Technology 164–165 (2005) 0, p. 1510-1518.

INTERNATIONAL STANDARDIZATION ORGANIZATION 2011

International Standardization Organization: ISO 10303-203:2011 Industrial automation systems and integration -- Product data representation and exchange -- Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies. International Standardization Organization 2011.

JIMÉNEZ et al. 2001

Jiménez, P.; Thomas, F.; Torras, C.: 3D collision detection: a survey. Computers & Graphics 25 (2001) 2, p. 269-285.

JONEJA & CHANG 1999

Joneja, A.; Chang, T.: Setup and fixture planning in automated process planning systems. IIE Transactions 31 (1999) 7, p. 653-665.

KAMARTHI et al. 2009

Kamarthi, S.; Bhole, N.; Zeid, A.: Investigating the design and development of truly agile flexible fixtures based on electrorheological fluids. International Journal of Rapid Manufacturing 1 (2009) 1, p. 99-110.

KANG & PENG 2008

Kang, X.; Peng, Q.: Fixture feasibility: methods and techniques for fixture planning. Computer-Aided Design and Applications 5 (2008) 1-4, p. 424-433.

KANG 2001

Kang, Y.: Computer-aided fixture design verification. Doctoral thesis, Manufacturing Engineering, Worcester Polytechnic Institute (2001). <http://www.wpi.edu/Pubs/ETD/Available/etd-0108102-163543/unrestricted/kang.pdf>

KANG et al. 2003a

Kang, Y.; Rong, Y.; Yang, J.: Computer-Aided Fixture Design Verification. Part 1. The Framework and Modelling. The International Journal of Advanced Manufacturing Technology 21 (2003a) 10, p. 827-835.

KANG et al. 2003b

Kang, Y.; Rong, Y.; Yang, J. A.: Geometric and Kinetic Model Based Computer-Aided Fixture Design Verification. Journal of Computing and Information Science in Engineering 3 (2003b) 3, p. 187-199.

KANG et al. 2003c

Kang, Y.; Rong, Y.; Yang, J. C.: Computer-Aided Fixture Design Verification. Part 2. Tolerance Analysis. The International Journal of Advanced Manufacturing Technology 21 (2003c) 10-11, p. 836-841.

KANG et al. 2003d

Kang, Y.; Rong, Y.; Yang, J. C.: Computer-Aided Fixture Design Verification. Part 3. Stability Analysis. The International Journal of Advanced Manufacturing Technology 21 (2003d) 10-11, p. 842-849.

KASHYAP & DEVRIES 1999

Kashyap, S.; DeVries, W.: Finite element analysis and optimization in fixture design. Structural and Multidisciplinary Optimization 18 (1999) 2, p. 193-201.

KAYA 2006

Kaya, N.: Machining fixture locating and clamping position optimization using genetic algorithms. Computers in Industry 57 (2006) 2, p. 112-120.

KONING & EIZENBERG 1981

Koning, H.; Eizenberg, J.: The language of the prairie: Frank Lloyd Wright's prairie houses. Environment and Planning B 8 (1981) 3, p. 295-323.

KOW et al. 2000

Kow, T. S.; Kumar, A. S.; Fuh, J. Y. H.: An Integrated Approach to Collision-Free Computer-Aided Modular Fixture Design. The International Journal of Advanced Manufacturing Technology 16 (2000) 4, p. 233-242.

KRISHNAKUMAR & MELKOTE 2000

Krishnakumar, K.; Melkote, S. N.: Machining fixture layout optimization using the genetic algorithm. International Journal of Machine Tools and Manufacture 40 (2000) 4, p. 579-598.

KRISHNAMURTI 1981

Krishnamurti, R.: The construction of shapes. Environment and Planning B: Planning and Design 8 (1981) 1, p. 5-40.

KRISHNAMURTI 1982

Krishnamurti, R. (Ed.): SGI: a shape grammar interpreter. Research Report, Center for configurational studies, The Open University, Milton Keynes, England (1982).

KRISHNAMURTI & STOUFFS 1993

Krishnamurti, R.; Stouffs, R.: Spatial Grammars: Motivation, Comparison, and New Results. In: Fifth International Conference on Computer-Aided Architectural Design Futures, Pittsburgh, PA, USA, 7-10 July 1993. Elsevier Science Publishers B.V.

LEE & HAYNES 1987

Lee, J.; Haynes, L.: Finite-element analysis of flexible fixturing system. Journal of Engineering for industry 109 (1987) 2, p. 134-139.

LI & MELKOTE 2001

Li, B.; Melkote, S.: Fixture clamping force optimisation and its impact on workpiece location accuracy. The International Journal of Advanced Manufacturing Technology 17 (2001) 2, p. 104-113.

LI & KUEN 2004

Li, I.; Kuen, L.: A set-based shape grammar interpreter, with thoughts on emergence. In: First International Conference on Design Computing and Cognition Workshop, 2004.

LI et al. 2002

Li, W.; Li, P.; Rong, Y.: Case-based agile fixture design. Journal of Materials Processing Technology 128 (2002) 1–3, p. 7-18.

LINDEMANN & BAUMBERGER 2006

Lindemann, U.; Baumberger, G.: Individualisierte Produkte. Springer 2006. ISBN: 3540255060.

MA et al. 1999

Ma, W.; Li, J.; Rong, Y.: Development of automated fixture planning systems. The International Journal of Advanced Manufacturing Technology 15 (1999) 3, p. 171-181.

MARIN & FERREIRA 2002

Marin, R. A.; Ferreira, P. M.: Optimal placement of fixture clamps: Minimizing the maximum clamping forces. Journal of manufacturing science and engineering 124 (2002) 3, p. 686-694.

MCCORMACK et al. 2004

McCormack, J. P.; Cagan, J.; Vogel, C. M.: Speaking the Buick language: capturing, understanding, and exploring brand identity with shape grammars. Design Studies 25 (2004) 1, p. 1-29.

MCKAY et al. 2012

McKay, A.; Chase, S.; Shea, K.; Chau, H. H.: Spatial grammar implementation: From theory to useable software. AI EDAM-Artificial Intelligence Engineering Design Analysis and Manufacturing 26 (2012) 2, p. 143.

MERVYN et al. 2006

Mervyn, F.; Senthil Kumar, A.; Nee, A. Y. C.: Fixture design information support for integrated design and manufacturing. International Journal of Production Research 44 (2006) 11, p. 2205-2219.

NEE & TAO 2004

Nee, A.; Tao, Z.: An advanced treatise on fixture design and planning. Singapore: World Scientific Pub Co Inc 2004. ISBN: 981-256-059-9.

OPEN CASCADE SAS 2013

Open Cascade SAS: Modeling Algorithms User's Guide 6.6.0 - Sweeping: Prism, Revolution and Pipe. <http://www.opencascade.org/getocc/download/occarchives/loadocc660/> (2013) April 09 2014. (Registration, download & installation required)

OPEN CASCADE SAS 2014a

Open Cascade SAS: Open CASCADE Technology, 3D modeling & numerical simulation <http://www.opencascade.org/> 2014 April 09.

OPEN CASCADE SAS 2014b

Open Cascade SAS: Salome - The Open Source Integration Platform for Numerical Simulations <http://www.salome-platform.org/> 2014 April 28.

PALMISANO et al. 2013

Palmisano, I.; Aranguren, M. E.; Rector, A.; Stevens, R.: OPPL 2 - Ontology Pre-Processing Language <http://oppl2.sourceforge.net/> 2014 28 August.

PEHLIVAN & SUMMERS 2008

Pehlivan, S.; Summers, J.: A review of computer-aided fixture design with respect to information support requirements. International Journal of Production Research 46 (2008) 4, p. 929-947.

PENG et al. 2011

Peng, G.; Chen, G.; Wu, C.; Xin, H.; Jiang, Y.: Applying RBR and CBR to develop a VR based integrated system for machining fixture design. Expert Systems with Applications 38 (2011) 1, p. 26-38.

PENG et al. 2010

Peng, G.; Wang, G.; Liu, W.; Yu, H.: A desktop virtual reality-based interactive modular fixture configuration design system. Computer-Aided Design 42 (2010) 5, p. 432-444.

PEROVIĆ 2013

Perović, B.: Vorrichtungen im Werkzeugmaschinenbau. Berlin: Springer 2013. ISBN: 3642327079.

PRATT 2001

Pratt, M. J.: Introduction to ISO 10303—the STEP Standard for Product Data Exchange. Journal of Computing and Information Science in Engineering 1 (2001) 1, p. 102-103.

PUGLIESE & CAGAN 2002

Pugliese, M. J.; Cagan, J.: Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar. Research in Engineering Design 13 (2002) 3, p. 139-156.

QT PROJECT HOSTING 2014

Qt Project Hosting: Qt Project <http://qt-project.org/> 2014 April 09.

RATCHEV et al. 2007

Ratchev, S.; Phuah, K.; Liu, S.: FEA-based methodology for the prediction of part–fixture behaviour and its applications. Journal of Materials Processing Technology 191 (2007) 1–3, p. 260-264.

RÍOS et al. 2005

Ríos, J.; Jiménez, J.; Pérez, J.; Vizán, A.; Menéndez, J.; Más, F.: KBE Application for the Design and Manufacture of HSM Fixtures. Acta Polytechnica 45 (2005) 3,

RONG & BAI 1997

Rong, Y.; Bai, Y.: Automated Generation of Fixture Configuration Design. Journal of Manufacturing Science and Engineering 119 (1997) 2, p. 208-219.

RONG et al. 2005

Rong, Y.; Huang, S.; Hou, Z.: Advanced computer-aided fixture design. San Diego, CA, USA: Academic Press 2005. ISBN: 0-12-594751-8.

ROY & LIAO 1999

Roy, U.; Liao, J.: Geometric reasoning for re-allocation of supporting and clamping positions in the automated fixture design system. IIE transactions 31 (1999) 4, p. 313-322.

ROY & LIAO 2002

Roy, U.; Liao, J.: Fixturing analysis for stability consideration in an automated fixture design system. Journal of manufacturing science and engineering 124 (2002) 1, p. 98-104.

SASS 2006

Sass, L.: A wood frame grammar: A generative system for digital fabrication. International Journal of Architectural Computing 4 (2006) 1, p. 51-67.

SASS 2008

Sass, L.: A physical design grammar: A production system for layered manufacturing machines. Automation in Construction 17 (2008) 6, p. 691-704.

SCHÜTZ et al. 2011

Schütz, D.; Schraufstetter, M.; Folmer, J.; Vogel-Heuser, B.; Gmeiner, T.; Shea, K.: Highly reconfigurable production systems controlled by real-time agents. In: Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on, Toulouse, France, 5-9 September 2011. p. 1-8. ISBN: 1946-0740.

SENTHIL KUMAR et al. 2000

Senthil Kumar, A.; Fuh, J.; Kow, T.: An automated design and assembly of interference-free modular fixture setup. Computer-aided design 32 (2000) 10, p. 583-596.

SENTHIL KUMAR et al. 1999

Senthil Kumar, A.; Subramaniam, V.; Seow, K.: Conceptual design of fixtures using genetic algorithms. The International Journal of Advanced Manufacturing Technology 15 (1999) 2, p. 79-84.

SHEA 2010

Shea, K.: The cognitive factory. Advanced Engineering Informatics 24 (2010) 3, p. 241-242.

SHEA & CAGAN 1997

Shea, K.; Cagan, J.: Innovative dome design: applying geodesic patterns with shape annealing. Artificial Intelligence for Engineering, Design, Analysis and Manufacturing 11 (1997) 05, p. 379-394.

SHEA et al. 2010

Shea, K.; Ertelt, C.; Gmeiner, T.; Ameri, F.: Design-to-fabrication automation for the cognitive machine shop. Advanced Engineering Informatics 24 (2010) 3, p. 251-268.

SHIRINZADEH 1993

Shirinzadeh, B.: Issues in the design of the reconfigurable fixture modules for robotic assembly. Journal of Manufacturing Systems 12 (1993) 1, p. 1-14.

SHIRINZADEH & TIE 1995

Shirinzadeh, B.; Tie, Y.: Experimental investigation of the performance of a reconfigurable fixturing system. The International Journal of Advanced Manufacturing Technology 10 (1995) 5, p. 330-341.

SIEBENALER & MELKOTE 2006

Siebenaler, S. P.; Melkote, S. N.: Prediction of workpiece deformation in a fixture system using the finite element method. International Journal of Machine Tools and Manufacture 46 (2006) 1, p. 51-58.

SIRIN et al. 2007

Sirin, E.; Parsia, B.; Grau, B. C.; Kalyanpur, A.; Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: science, services and agents on the World Wide Web 5 (2007) 2, p. 51-53.

STANFORD CENTER FOR BIOMEDICAL INFORMATICS RESEARCH 2011

Stanford Center for Biomedical Informatics Research: Protégé <http://protege.stanford.edu/> 2014 March 10.

STARLING & SHEA 2002

Starling, A. C.; Shea, K.: A clock grammar: the use of a parallel grammar in performance-based mechanical synthesis. In: 14th International Conference on Design Theory and Methodology, Integrated Systems Design, and Engineering Design and Culture, Montreal, Quebec, Canada, 2002. ASME 2002, p. 287-296.

STINY 1977

Stiny, G.: Ice-ray: a note on the generation of Chinese lattice designs. Environment and Planning B 4 (1977) 1, p. 89-98.

STINY 1980a

Stiny, G.: Introduction to shape and shape grammars. Environment and Planning B: Planning and Design 7 (1980a) 3, p. 343-351.

STINY 1980b

Stiny, G.: Kindergarten grammars designing with Froebel's building gifts. Environment and Planning B: Planning and Design 7 (1980b) p. 409-462.

STINY 1982

Stiny, G.: Spatial relations and grammars. Environment and Planning B: Planning and Design 9 (1982) 1, p. 113-114.

STINY & GIPS 1972

Stiny, G.; Gips, J.: Shape grammars and the generative specification of painting and sculpture. Information processing 71 (1972) 1460-1465,

STINY & MITCHELL 1978

Stiny, G.; Mitchell, W. J.: The palladian grammar. Environment and Planning B 5 (1978) 1, p. 5-18.

STUCKENSCHMIDT 2009

Stuckenschmidt, H.: Ontologien: Konzepte, Technologien und Anwendungen. Berlin: Springer 2009. ISBN: 9783540793304. (Informatik im Fokus).

SURE & STUDER 2002

Sure, Y.; Studer, R. (Eds.): On-To-Knowledge Methodology - Final Version. Institute AIFB, University of Karlsruhe, Karlsruhe (2002).

TRAPPEY & LIU 1993

Trappey, A.; Liu, C.: Automated fixture configuration using projective geometry approach. The International Journal of Advanced Manufacturing Technology 8 (1993) 5, p. 297-304.

TRAPPEY & LIU 1990

Trappey, J.; Liu, C.: A literature survey of fixture-design automation. The International Journal of Advanced Manufacturing Technology 5 (1990) 3, p. 240-255.

USCHOLD & GRUNINGER 1996

Uschold, M.; Gruninger, M.: Ontologies: Principles, methods and applications. The Knowledge Engineering Review 11 (1996) 02, p. 93-136.

VAN AART et al. 2002

van Aart, C.; Pels, R.; Caire, G.; Bergenti, F.: Creating and using ontologies in agent communication. In: Second International Workshop on Ontologies in Agent Systems (OAS 2002) , held at the 1st International Conference on Autonomous Agents & Multiagent Systems, Bologna, Italy, 15-16 July 2002.

WALLACK & CANNY 1996

Wallack, A. S.; Canny, J. F.: Modular fixture design for generalized polyhedra. In: 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, USA, April 22-28 1996. IEEE p. 830-837. ISBN: 0780329880.

WANG et al. 2010

Wang, H.; Rong, K. Y.; Li, H.; Shaun, P.: Computer aided fixture design: Recent research and trend. Computer-aided design (2010)

WANG & RONG 2008

Wang, H.; Rong, Y.: Case based reasoning method for computer aided welding fixture design. Computer-Aided Design 40 (2008) 12, p. 1121-1132.

WANG 2002

Wang, M. Y.: Tolerance analysis for fixture layout design. Assembly Automation 22 (2002) 2, p. 153-162.

WILLY et al. 1995

Willy, A.; Sadler, J. P.; Schraft, R. D.: Automated fixture design. The International Journal of Advanced Manufacturing Technology 10 (1995) 1, p. 27-35.

WU et al. 1998a

Wu, Y.; Rong, Y.; Ma, W.; LeClair, S.: Automated modular fixture design: geometric analysis. Robotics and Computer-Integrated Manufacturing 14 (1998a) 1, p. 1-15.

WU et al. 1998b

Wu, Y.; Rong, Y.; Ma, W.; LeClair, S.: Automated modular fixture planning: Accuracy, clamping, and accessibility analyses. Robotics and Computer-Integrated Manufacturing 14 (1998b) 1, p. 17-26.

ZAEH et al. 2012

Zaeh, M. F.; Ostgathe, M.; Geiger, F.; Reinhart, G.: Adaptive Job Control in the Cognitive Factory. In: ElMaraghy, H. A. (Ed.): Enabling Manufacturing Competitiveness and Economic Sustainability. Berlin: Springer 2012, p. 10-17. ISBN: 978-3-642-23859-8.

ZAEH et al. 2010

Zaeh, M. F.; Reinhart, G.; Ostgathe, M.; Geiger, F.; Lau, C.: A holistic approach for the cognitive control of production systems. Advanced Engineering Informatics 24 (2010) 3, p. 300-307.

ZHENG 2005

Zheng, Y.: Finite element analysis for fixture stiffness. Doctoral thesis, Manufacturing Engineering, Worcester Polytechnic Institute. (2005). <http://www.wpi.edu/Pubs/ETD/Available/etd-050505-130954/unrestricted/zheng-dissertation.pdf> -

ZHENG & CHEW 2009

Zheng, Y.; Chew, C.-M.: Efficient Procedures for Form-Closure Grasp Planning and Fixture Layout Design. Journal of Manufacturing Science and Engineering 131 (2009) 4, p. 041010.

ZHENG et al. 2005

Zheng, Y.; Rong, Y.; Hou, Z.: A finite element analysis for stiffness of fixture units. Journal of Manufacturing Science and Engineering 127 (2005) p. 429.

ZHU 2009

Zhu, X. a. H., D.: Optimality Criteria for Fixture Layout Design: A Comparative Study. IEEE Transactions on Automation Science and Engineering 6 (2009) 4, p. 12.

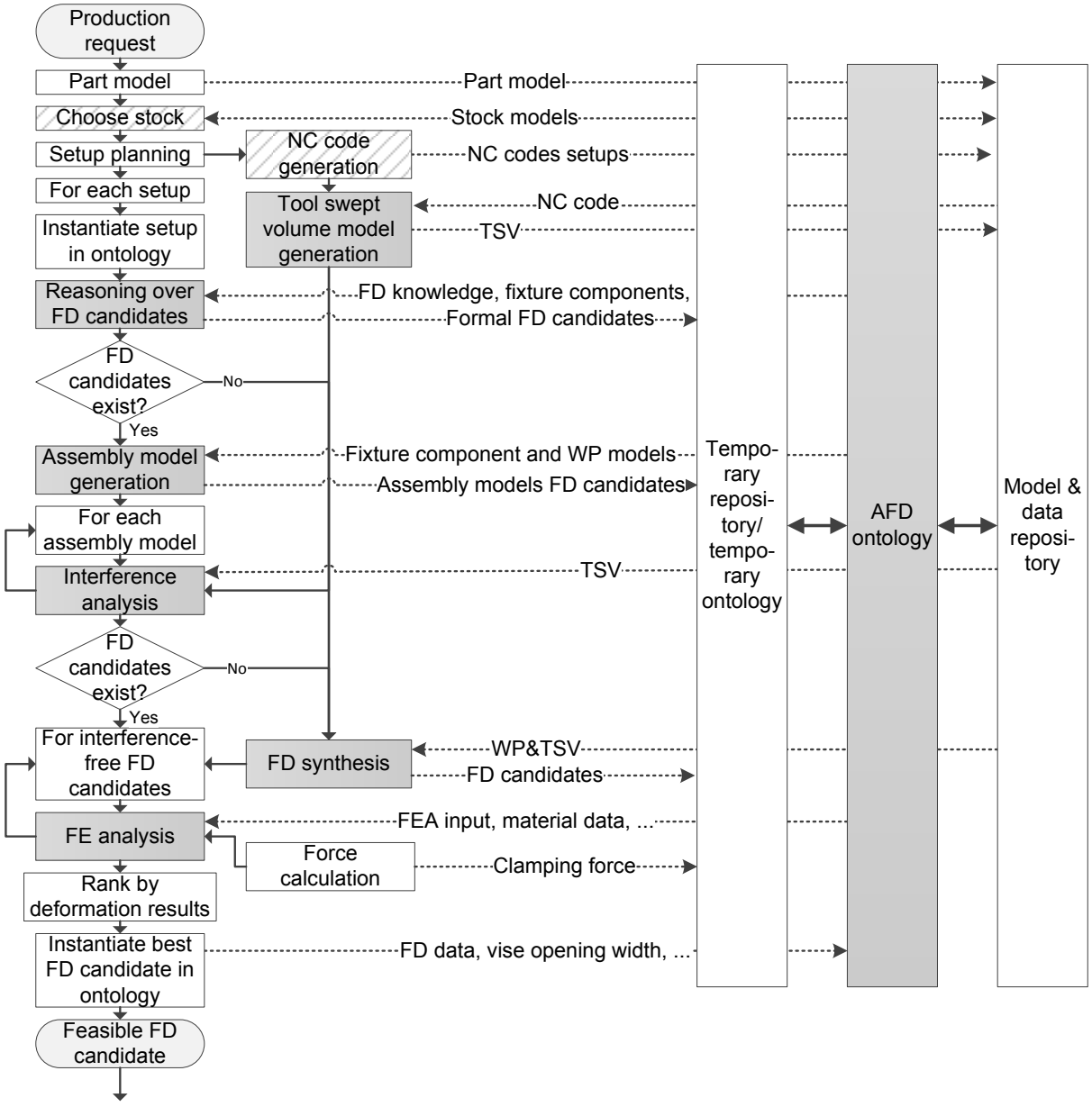
## Software Resources

Fixture Design Ontologies: https://ontohub.org/repositories/fixture-design-ontologies

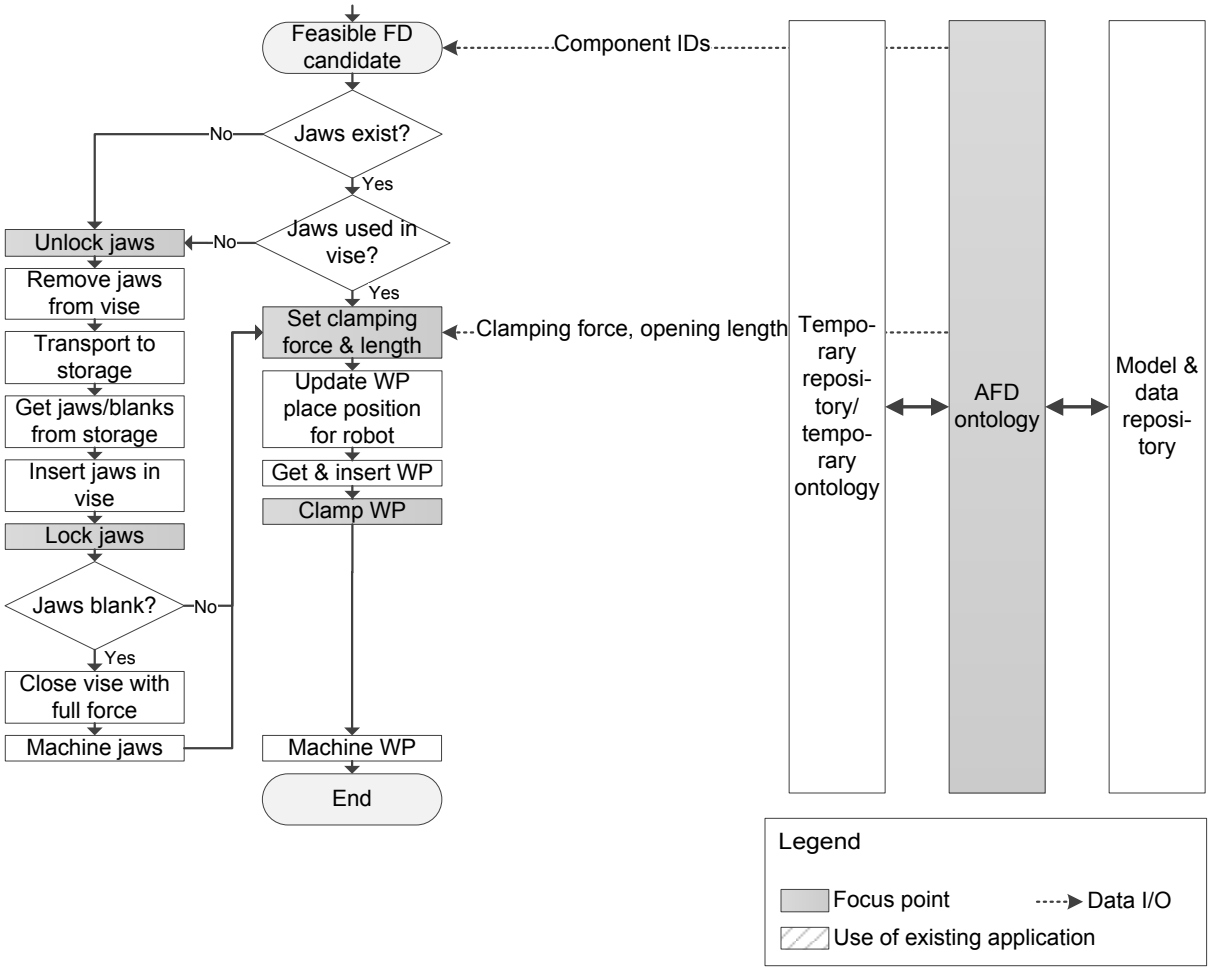
Jaw Set Grammar: http://sourceforge.net/projects/spapper/files/

# 10. Appendix

## 10.1 Extended process scheme for the AFD system



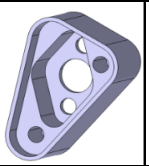
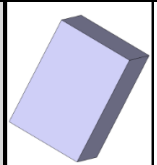
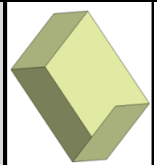
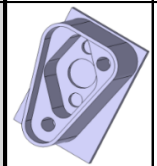
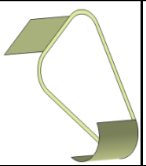
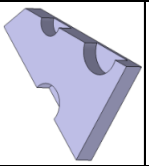
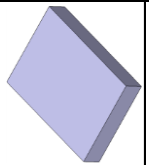
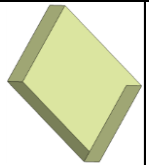
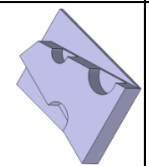
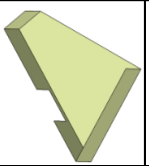
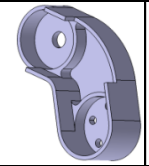
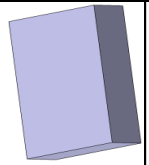
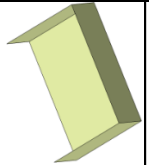
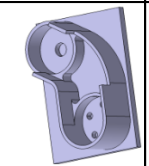
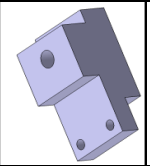
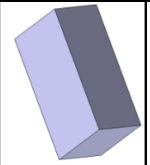
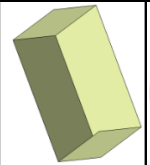
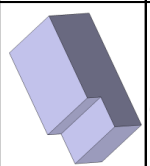
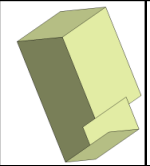
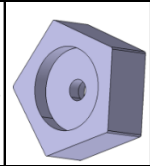
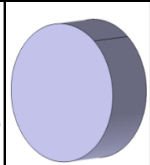
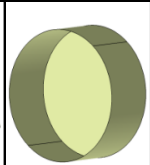
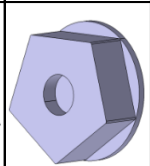
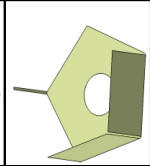
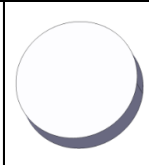
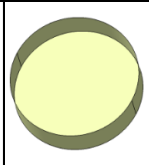
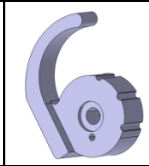
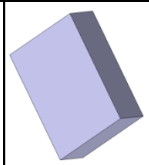
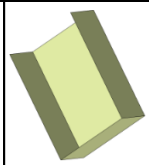
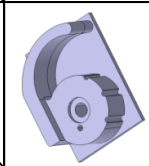
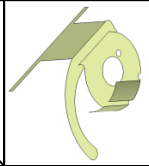

Part 1 of 2


Feasible FD candidate
Component IDs
Jaws exist?
No
Yes
Jaws used in vise?
No
Yes
Unlock jaws
Remove jaws from vise
Transport to storage
Get jaws/blanks from storage
Insert jaws in vise
Lock jaws
Jaws blank?
No
Yes
Close vise with full force
Machine jaws
Set clamping force & length
Clamping force, opening length
Update WP place position for robot
Get & insert WP
Clamp WP
Machine WP
End
Tempo-rary reposi-tory/ tempo-rary ontology
AFD ontology
Model & data reposi-tory
Legend
Focus point
Data I/O
Use of existing application

Part 2 of 2

## 10.2 Case-study parts

List of 14 case-study parts with intermediate workpiece states and contact surfaces of each setup – part 1 of 2

| Name | Part ID | Dim. blank | # of setups | Part | Blank | Setup 1 | Intermediate_ 1 | Setup 2 | Intermediate_ 2 | Setup 3 | Source |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Allied Signal ACIS09 | 9001 | 110x 75x 45mm | 2 |  |  |  |  |  | | | Drexel Design Repository |
| Angled Socket | 9015 | 98x 70x 18mm | 2 |  |  |  |  |  | | | CogMaSh |
| Cap Arm | 9017 | 126x 90x 35mm | 2 |  |  |  |  |  | | | Drexel Design Repository |
| Cube_ CS75x 40x30 | 9005 | 75x 40x 30mm | 2 |  |  |  |  |  | | | CogMaSh |
| Penta-gon | 9016 | D68x 35mm | 2 |  |  |  |  |  | | | CogMaSh |
| Cylinder D60_ H15 | 9008 | D60x 16mm | 1 |  |  |  | | | | | CogMaSh |
| Lever | 9019 | 123x 88x 45mm | 2 |  |  |  |  |  | | | Drexel Design Repository |

List of 14 case-study parts with intermediate workpiece states and contact surfaces of each setup – part 2 of 2

| Name | Part ID | Dim. blank | # of setups | Part | Blank | Setup 1 | Intermediate_ 1 | Setup 2 | Intermediate_ 2 | Setup 3 | Source |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Cyl_ Plane | 9017 | 104x 60x 25mm | 2 | | | | | | | | CogMaSh |
| Half-Butterfly | 9009 | 70x 50x 22mm | 2 | | | | | | | | CogMaSh |
| Holder | 9010 | 120x 70x 20mm | 2 | | | | | | | | CogMaSh |
| Rec_FR | 9018 | 75x 54x 15mm | 2 | | | | | | | | Drexel Design Repository |
| Cylinder D40_ H70 | 9007 | D40x 70mm | 2 | | | | | | | | CogMaSh |
| Winged Flange | 9011 | 130x 100x 13mm | 3 | | | | | | | | CogMaSh |
| Socket | 9013 | D100x 75mm | 3 | | | | | | | | Drexel Design Repository |

## 10.3 Evaluation Data for the AFD Ontology

### 10.3.1 Data of Jaws Instantiated

| Flex Vise Jaw | has ID | hasJaw Shape | has Step Depth | has Step Width | has Step Length | hasGeometric Entity/ hasContactEntity | Setup function | (Surface) has Surface Shape |
|---|---|---|---|---|---|---|---|---|
| FVJ_St01 | 3001 | Step | 2 | 100 | 4 | Solid_3001 | | |
| | | | | | | Surface_3001_0 | Support | Planar |
| | | | | | | Surface_3001_1 | Clamp | Planar |
| FVJ_St02 | 3002 | Step | 3 | 100 | 7 | Solid_3002 | | |
| | | | | | | Surface_3002_0 | Support | Planar |
| | | | | | | Surface_3002_1 | Clamp | Planar |
| FVJ_St03 | 3003 | Step | 12 | 100 | 3 | Solid_3003 | | |
| | | | | | | Surface_3003_0 | Support | Planar |
| | | | | | | Surface_3003_1 | Clamp | Planar |
| FVJ_St04 | 3004 | Step | 13 | 100 | 11 | Solid_3004 | | |
| | | | | | | Surface_3004_0 | Support | Planar |
| | | | | | | Surface_3004_1 | Clamp | Planar |
| FVJ_StR_Pos1 | 1001 | StepRestPos | 2 | 60 | 6 | Solid_1001 | | |
| | | | | | | Surface_1001_0 | PrimLoc | Planar |
| | | | | | | Surface_1001_1 | SecLoc | Planar |
| | | | | | | Surface_1001_2 | TertLoc | Cylindrical |
| FVJ_StR_Pos2 | 1002 | StepRestPos | 3 | 79 | 9 | Solid_1002 | | |
| | | | | | | Surface_1002_0 | PrimLoc | Planar |
| | | | | | | Surface_1002_1 | SecLoc | Planar |
| | | | | | | Surface_1002_2 | TertLoc | Cylindrical |
| FVJ_StR_Neg1 | 1003 | StepRestNeg | 12 | 84 | 7 | Solid_1003 | | |
| | | | | | | Surface_1003_0 | PrimLoc | Planar |
| | | | | | | Surface_1003_1 | SecLoc | Planar |
| | | | | | | Surface_1003_2 | TertLoc | Cylindrical |
| FVJ_StR_Neg2 | 1004 | StepRestNeg | 3 | 87 | 12 | Solid_1004 | | |
| | | | | | | Surface_1004_0 | PrimLoc | Planar |
| | | | | | | Surface_1004_1 | SecLoc | Planar |
| | | | | | | Surface_1004_2 | TertLoc | Cylindrical |
| FVJ_V01 | 2001 | V | 12 | -- | 10 | Solid_2001 | | |
| | | | | | | Surface_2001_0 | PrimLoc | Planar |
| | | | | | | Surface_2001_1 | SecLoc | Planar |
| | | | | | | Surface_2001_2 | TertLoc | Planar |
| FVJ_V02 | 2002 | V | 13 | -- | 13 | Solid_2002 | | |
| | | | | | | Surface_2002_0 | PrimLoc | Planar |
| | | | | | | Surface_2002_1 | SecLoc | Planar |
| | | | | | | Surface_2002_2 | TertLoc | Planar |
| FVJ_V03 | 2003 | V | 2 | -- | 12 | Solid_2003 | | |
| | | | | | | Surface_2003_0 | PrimLoc | Planar |
| | | | | | | Surface_2003_1 | SecLoc | Planar |
| | | | | | | Surface_2003_2 | TertLoc | Planar |

Data jaws part 1 of 2

| (Surface) has Outward Normal | (Vector) has Direction | (Surface) has Point | (Point) has Coordinate | (Solid) has ModelPath | (Solid) has Labeled ModelPath |
|---|---|---|---|---|---|
| | | | | C://AnyPath/ 3001.step | C://AnyPath/ 3001_labeled. FCStd |
| Vector_3001_0 | (0,0,-1) | Point_3001_0 | (x,y,z) | | |
| Vector_3001_1 | (-1,0,0) | Point_3001_1 | (x,y,z) | | |
| | | | | C://AnyPath/ 3002.step | C://AnyPath/ 3002_labeled. FCStd |
| Vector_3002_0 | (0,0,-1) | Point_3002_0 | (x,y,z) | | |
| Vector_3002_1 | (-1,0,0) | Point_3002_1 | (x,y,z) | | |
| | | | | C://AnyPath/ 3003.step | C://AnyPath/ 3003_labeled. FCStd |
| Vector_3003_0 | (0,0,-1) | Point_3003_0 | (x,y,z) | | |
| Vector_3003_1 | (-1,0,0) | Point_3003_1 | (x,y,z) | | |
| | | | | C://AnyPath/ 3004.step | C://AnyPath/ 3004_labeled. FCStd |
| Vector_3004_0 | (0,0,-1) | Point_3004_0 | (x,y,z) | | |
| Vector_3004_1 | (-1,0,0) | Point_3004_1 | (x,y,z) | | |
| | | | | C://AnyPath/ 1001.step | C://AnyPath/ 1001_labeled. FCStd |
| Vector_1001_0 | (0,0,-1) | Point_1001_0 | (x,y,z) | | |
| Vector_1001_1 | (1,0,0) | Point_1001_1 | (x,y,z) | | |
| Vector_1001_2 | (0,1,0) | Point_1001_2 | (x,y,z) | | |
| | | | | C://AnyPath/ 1002.step | C://AnyPath/ 1002_labeled. FCStd |
| Vector_1002_0 | (0,0,-1) | Point_1002_0 | (x,y,z) | | |
| Vector_1002_1 | (1,0,0) | Point_1002_1 | (x,y,z) | | |
| Vector_1002_2 | (0,1,0) | Point_1002_2 | (x,y,z) | | |
| | | | | C://AnyPath/ 1003.step | C://AnyPath/ 1003_labeled. FCStd |
| Vector_1003_0 | (0,0,-1) | Point_1003_0 | (x,y,z) | | |
| Vector_1003_1 | (1,0,0) | Point_1003_1 | (x,y,z) | | |
| Vector_1003_2 | (0,-1,0) | Point_1003_2 | (x,y,z) | | |
| | | | | C://AnyPath/ 1004.step | C://AnyPath/ 1004_labeled. FCStd |
| Vector_1004_0 | (0,0,-1) | Point_1004_0 | (x,y,z) | | |
| Vector_1004_1 | (1,0,0) | Point_1004_1 | (x,y,z) | | |
| Vector_1004_2 | (0,-1,0) | Point_1004_2 | (x,y,z) | | |
| | | | | C://AnyPath/ 2001.step | C://AnyPath/ 2001_labeled. FCStd |
| Vector_2001_0 | (0,0,-1) | Point_2001_0 | (x,y,z) | | |
| Vector_2001_1 | (0.868243, 0.496139, 0) | Point_2001_1 | (x,y,z) | | |
| Vector_2001_2 | (0.868243,-0.496139, 0) | Point_2001_2 | (x,y,z) | | |
| | | | | C://AnyPath/ 2002.step | C://AnyPath/ 2002_labeled. FCStd |
| Vector_2002_0 | (0,0,-1) | Point_2002_0 | (x,y,z) | | |
| Vector_2002_1 | (0.933346, 0.358979, 0) | Point_2002_1 | (x,y,z) | | |
| Vector_2002_2 | (0.933346, -0.358979, 0) | Point_2002_2 | (x,y,z) | | |
| | | | | C://AnyPath/ 2003.step | C://AnyPath/ 2003_labeled. FCStd |
| Vector_2003_0 | (0,0,-1) | Point_2003_0 | (x,y,z) | | |
| Vector_2003_1 | (-0.911922, -0.410365, 0) | Point_2003_1 | (x,y,z) | | |
| Vector_2003_2 | (-0.911922, 0.410365, 0) | Point_2003_2 | (x,y,z) | | |

Data jaws part 2 of 2

## 10.3.2 Data of Workpieces Instantiated

| Part | Work-piece | hasID | hasGeometricEntity/hasSetupEntity/ hasCoM | Setup function | (Surface) hasSurf-aceShape |
|---|---|---|---|---|---|
| AS_ACIS09 | 9001_0 | 9001_0 | Solid_9001_0 | | |
| | | | Surface_9001_0_0 | PrimLoc, Supp | Planar |
| | | | Surface_9001_0_1 | SecLoc | Planar |
| | | | Surface_9001_0_2 | TertLoc | Planar |
| | | | Surface_9001_0_3 | Clamp | Planar |
| | | | Surface_9001_0_4 | Active | Planar |
| | | | Point_9001_0_CoM | -- | -- |
| | 9001_1 | 9001_1 | Solid_9001_1 | | |
| | | | Surface_9001_1_0 | PrimLoc, Supp | Planar |
| | | | Surface_9001_1_1 | SecLoc | Cylindrical |
| | | | Surface_9001_1_2 | Clamp | Planar |
| | | | Surface_9001_0_3 | Active | Planar |
| | | | Point_9001_1_CoM | -- | -- |
| Cap Arm | 9017_0 | 9017_0 | Solid_9017_0 | | |
| | | | Surface_9017_0_0 | PrimLoc, Supp | Planar |
| | | | Surface_9017_0_1 | SecLoc | Planar |
| | | | Surface_9017_0_2 | TertLoc | Planar |
| | | | Surface_9017_0_3 | Clamp | Planar |
| | | | Surface_9017_0_4 | Active | Planar |
| | | | Point_9017_0_CoM | -- | -- |
| | 9017_1 | 9017_1 | Solid_9017_1 | | |
| | | | Surface_9017_1_0 | PrimLoc | Planar |
| | | | Surface_9017_1_1 | SecLoc | Cylindrical |
| | | | Surface_9017_1_2 | Clamp | Cylindrical |
| | | | Surface_9017_1_3 | Supp | Planar |
| | | | Surface_9017_1_4 | Active | Planar |
| | | | Point_9017_1_CoM | -- | -- |

Data workpieces part 1 of 2

| (Surface) has OutwardNormal | (Vector) has Direction | (Surface) hasPoint | (Point) has Coordinate | (Solid) has ModelPath | (Solid) hasLabeledModelPath |
|---|---|---|---|---|---|
| | | | | C://AnyPath/9001_0.step | C://AnyPath/9001_0_labeled.FCStd |
| Vector_9001_0_0 | 0,0,1 | Point_9001_0_0 | (x,y,1044) | | |
| Vector_9001_0_1 | (x,y,z) | Point_9001_0_1 | (1000,y,z) | | |
| Vector_9001_0_2 | 0,-1,0 | Point_9001_0_2 | (x,962.5,z) | | |
| Vector_9001_0_3 | (x,y,z) | Point_9001_0_3 | (1110,y,z) | | |
| Vector_9001_0_4 | (x,y,z) | Point_9001_0_4 | (x,y,z) | | |
| -- | -- | -- | (x,1000,z) | | |
| | | | | C://AnyPath/9001_1.step | C://AnyPath/9001_1_labeled.FCStd |
| Vector_9001_1_0 | 0,0,1 | Point_9001_1_0 | (x,y,1040) | | |
| Vector_9001_1_1 | (x,y,z) | Point_9001_1_1 | (1000,y,z) | | |
| Vector_9001_1_2 | (x,y,z) | Point_9001_1_2 | (1106,y,z) | | |
| Vector_9001_0_3 | (x,y,z) | Point_9001_0_3 | (x,y,z) | | |
| -- | -- | -- | (x,999.49,z) | | |
| | | | | C://AnyPath/9017_0.step | C://AnyPath/9017_0_labeled.FCStd |
| Vector_9017_0_0 | 0,0,1 | Point_9017_0_0 | (x,y,505) | | |
| Vector_9017_0_1 | (x,y,z) | Point_9017_0_1 | (438,y,z) | | |
| Vector_9017_0_2 | 0,1,0 | Point_9017_0_2 | (x,528,z) | | |
| Vector_9017_0_3 | (x,y,z) | Point_9017_0_3 | (569,y,z) | | |
| Vector_9017_0_4 | (x,y,z) | Point_9017_0_4 | (x,y,z) | | |
| -- | -- | -- | (x,465,z) | | |
| | | | | C://AnyPath/9017_1.step | C://AnyPath/9017_1_labeled.FCStd |
| Vector_9017_1_0 | 0,0,1 | Point_9017_1_0 | (x,y,530) | | |
| Vector_9017_1_1 | (x,y,z) | Point_9017_1_1 | (475.2,y,z) | | |
| Vector_9017_1_2 | (x,y,z) | Point_9017_1_2 | (603.3,y,z) | | |
| Vector_9017_1_3 | 0,0,1 | Point_9017_1_3 | (x,y,520) | | |
| Vector_9017_1_4 | (x,y,z) | Point_9017_1_4 | (x,y,z) | | |
| -- | -- | -- | (x,496.85,z) | | |

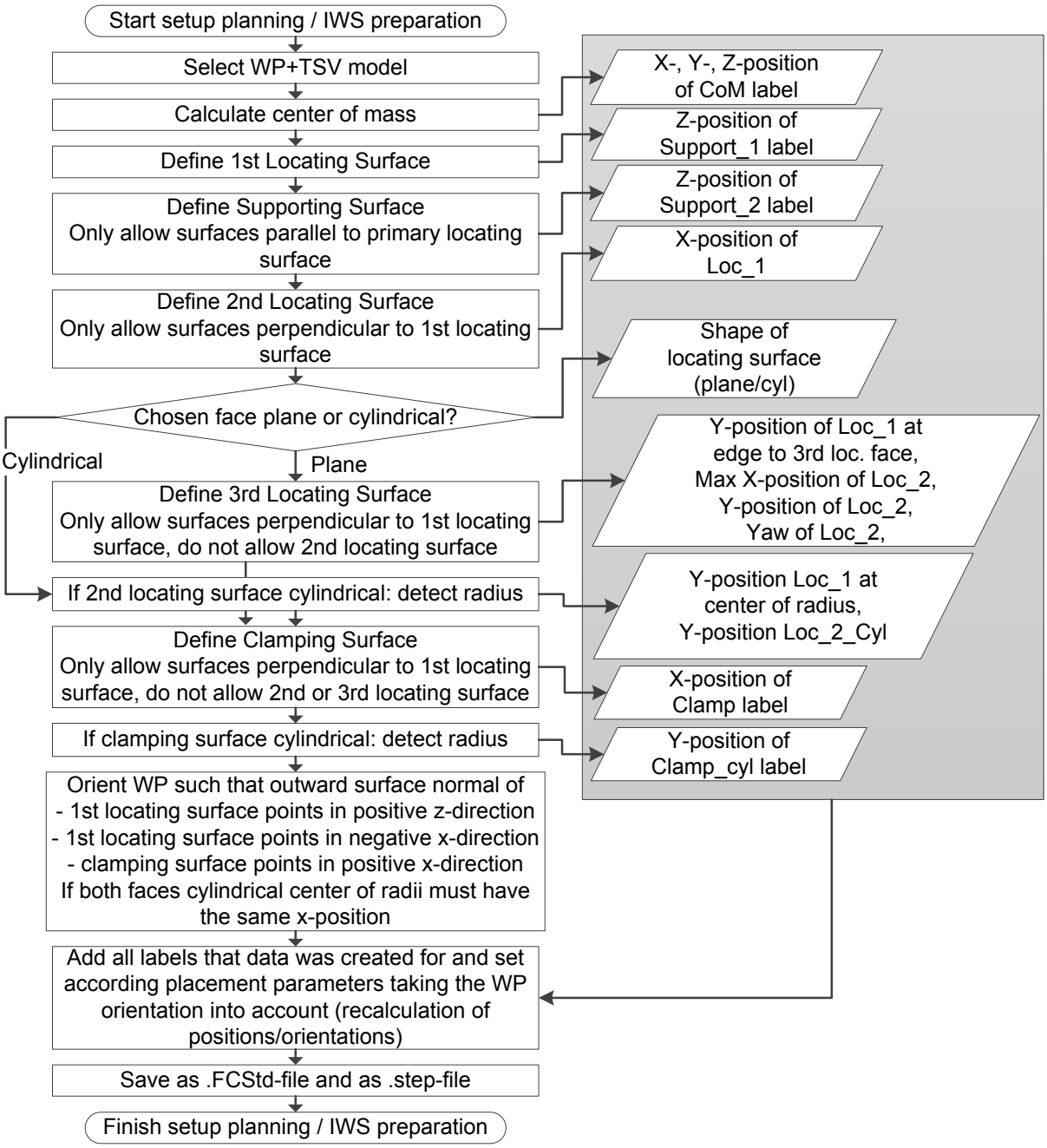
Data workpieces part 2 of 2

## 10.4 Cutting tools in EMCO Concept Mill 155

Data of the cutting tools installed on the EMCO Concept Mill 155, the 3-axis milling machine of the CogMaSh project that the flexible vise is installed in. Tool slot T7 is not used.

| TOOL_NAME | TOOL_# | NOMINAL_DIAM | OVERALL_LGTH | LENGTH | CUT_LENGTH | BODY_DIAM | ENTRY_DIAM | CUT_ANGLE |
|---|---|---|---|---|---|---|---|---|
| String | Integer | mm | mm | mm | mm | mm | mm | deg |
| T1 Face Mill D40 | 1 | 40 | 48 | 20 | 20 | 27 | -- | 90 |
| T2 End Mill D10 | 2 | 10 | 32 | 32 | 20 | 10 | -- | 90 |
| T3 Conical Mill Tool D16 | 3 | 16 | 32 | 0 | 5 | 10 | 6 | 90 |
| T4 End Mill D6 | 4 | 6 | 20 | 20 | 16 | 6 | -- | 90 |
| T5 Conical Mill Tool D10 | 5 | 10 | 25 | 1 | 5 | 10 | 0 | 90 |
| T6 End Mill D10 | 6 | 10 | 31 | 31 | 23 | 10 | -- | 90 |
| -- | 7 | -- | -- | -- | -- | -- | -- | 90 |
| T8 End Mill D6 | 8 | 6 | 23 | 23 | 10 | 6 | -- | 90 |
| T9 End Mill D15 | 9 | 15 | 43 | 27 | 27 | 12 | -- | 90 |
| T10 Drill D6 | 10 | 6 | 70 | 38 | 30 | 6 | -- | 120 |

## 10.5 Model preparation process – orientation and labeling




Process scheme for the IWS preparation and manual setup planning – the combined WP&TSV model is imported, the contact surfaces are defined, the model oriented according to the grammar requirements and the IWS labels generated and placed;

## 10.6 Jaw set design requirements and constraints

**Equivalent to Table 5-2 List of requirements for the development of the jaw set grammar that must or should (fixed/flexible) be embedded in the designs synthesized by the grammar**

| Gmeiner TU München | | List of Requirements Development of a jaw set grammar | Page 1/1 | |
|---|---|---|---|---|
| **Nr.** | **Fixed/ Flexible** | **Requirement** | **Value** | **Unit** |
| 1 | Fix | Accurate locating for generalized polyhedral WP shapes | -- | -- |
| 2 | Fix | Ensure WP stability during machining | -- | -- |
| 3 | Fix | No alteration of functional features of jaws | -- | -- |
| 4 | Fix | No tool-fixture interferences | -- | -- |
| 5 | Fix | Allow for stable (un-)loading of vise | -- | -- |
| 6 | Fix | Ensure machinability of designs with existing cutting tools | -- | -- |
| 7 | Fix | Ensure machinability of designs with machining process | -- | -- |
| 8 | Fix | Different supporting surfaces at anvil and slide possible | -- | -- |
| 9 | Fix | Compliance with given raw jaw model and dimensions | 28x100x40 | mm |
| 10 | Flex | Maximum opening length restriction of vise | 120 | mm |
| 11 | Fix | Minimum opening length restriction of vise | 4 | mm |
| 12 | Flex | Minimum height for locating/clamping surfaces of jaws | 2 | mm |
| 13 | Flex | Opening angel of V-blocks | 120°-140° | deg |
| 14 | Flex | Minimum remaining material thickness in x-direction | 2 | mm |
| 15 | Flex | Minimum remaining material thickness in y-direction | 10 | mm |
| 16 | Flex | Minimum remaining material thickness in z -direction | 5 | mm |

### 10.6.1 Details on requirements

**Requirement no. 1:**

Accurate locating of the workpieces must be guaranteed with all synthesized jaw sets. This includes that a correct locating principle is chosen as basis for the jaw set design and that the

design variables are set such that the locating contacts between the jaws and the WP are established correctly. For step-rest-shaped locating jaws with rounded locator in y-direction this means the establishment of exactly two surface contacts (primary supporting and primary locating) and one line contact (secondary locating). For V-shaped locating jaws this means the establishment of exactly one surface contact (primary supporting) and two line contact (primary locating) at the sloped surfaces of the V-block.

**Requirement no. 2:**

The WP must be clamped such that the clamping forces and reaction forces at the contacts can withstand all machining forces without a displacement of the WP (beyond the deformation of the fixture). This requires that a clamping scheme that matches the locating scheme is used, i.e. that the clamping jaws are a shape that matches the locating jaw shape, and that the clamping force is set to withstand the machining forces.

**Requirement no. 3:**

The functional features of the blank jaws which are required for handling the jaws in the FMS and attaching the jaws to the vise must not be altered by the machining of the jaw set design. This is discussed in Section 5.3.1.

**Requirement no. 4:**

The resulting jaw set designs must not collide with the total tool swept volume model (TTSV), if such a model exists in the initial working shape. The TTSV represents the 3D volume that the tool(s) require(s) for the machining of the given setup. It is generated and added to the WP model as part of the interference analysis, as described in Section 6.1. The jaws are not allowed to collide with this volume as this would result in an unwanted interference between the cutting tool(s) and the jaws.

**Requirement no. 5:**

The jaws have to be shaped such that the WP can be loaded into the vise and so that it does not move or displace when the vise is not engaged. When a WP is loaded in the vise, it is disengaged, causing the opening length of the vise to be slightly bigger than the clamping width of the WP. This way, the handling robot can easily load and unload the vise. Once having placed the WP in the vise, the robot lets go of the WP. The WP then rests on the anvil and slide jaw supporting surfaces. The jaws must, therefore, be shaped in a way that allows for the stable positioning of the WP under the gravity forces and in absence of any other forces like the clamping or robot grasping force.

**Requirement no. 6&7:**

All jaw set design must be machinable with the given machine tool and its tool approach direction, the available cutting tools and the predefined machining process, where the blank jaws are pressed against each other during the machining process. These factors play an important role on the volume that can be removed from the blank jaws. For the given scenario the milling of the jaws should always be done with the smallest cutting tool, due to the restricted nature of the machinable volume.

**Requirement no. 8:**

Two different plane and parallel WP surfaces with the same outward normal direction can be used for supporting the WP, called the primary (locating side) and the secondary (clamping side) supporting surface. This requires supporting surfaces on both the locating and the clamping jaw, which are required anyway to meet requirement no.5. The supporting surfaces of the jaws can, thus, be at different z-positions. The maximum allowed z-difference between the WP supporting surfaces is defined by the z-depth of the machinable volume, taking into account that a minimum clamping surface height needs to remain at each jaw.

**Requirement no. 9:**

The shape and dimensions of the blank jaws for the flexible vise that are the basis for all jaw sets are fixed.

**Requirement no. 10&11:**

The flexible vise has a maximum opening length defined by the device structure. The capabilities of the vise also require a limit on the minimum opening length as smaller values cannot be securely set by the pneumatic drive and the control system.

**Requirement no. 12:**

A user set requirement is that the locating and clamping surfaces must at least have a minimum height to provide some contact area/line length between the jaws and the WP.

**Requirement no. 13:**

The opening angle for all V-block jaws should be within a given range. The values are standard for V-block locators and were user-defined. The opening angle is defined via the ratio of wedge length to wedge width.

**Requirement no. 14-16:**

A minimum material thickness must remain when cutouts are being made from the jaw in all directions, to ensure that the contact entities can withstand the reaction forces caused by the clamping and machining forces. The values are user defined.

## 10.6.2 Details on design constraints

Here the exact geometric constraints imposed by the requirements on the design parameters are discussed. First the detailed values for the removable volume of the all jaws are defined, as presented in Section 5.3.1. In the x-direction 2 mm of material in front of the functional features of the jaws should remain unaltered. In combination with the dimensions of the functional features in x-direction and the maximum length of the blank jaws this leads to a removable length of 12 mm. In y-direction, the minimum remaining width should be 10 mm, either on the left or on the right of the jaw, leaving a removable width of 100 mm-10 mm=90 mm. And in z-direction a height of 5 mm should remain, resulting in a possible removable depth of 40 mm-5 mm=35 mm. However, the cutting tools available in the milling machine the vise is installed in also have to be taken into account for the definition of the removable volume. The smallest cutting tool is a Ø6 mm end-mill cutter, as described in Appendix 10.3, with a free length of 23 mm. Leaving 1 mm of length as safety the cutter length sets the maximum cutout depth $z_{step}$ for all jaws to 22 mm, a tighter constraint than the previously determined 35 mm.

Next, the effect of the requirements on the design variables and parameter interdependencies of each basic jaw shape are discussed.

### Constraints on step-jaw parameters

For St-jaws, which can only be used as clamping jaws, only the length ($x_{step}$) and depth ($z_{step}$) of the step are variable. The maximum values for both are defined by the removable volume, as discussed before. The minimum values for both are 2 mm based on the need for a minimum supporting surface (requirement no.5) and minimum clamping height (requirement no.12). However, the minimum value for the step length $x_{step}$ can also be bigger, dependent on the step length of the locating jaw. The sum of both $x_{step}$ values has to be bigger than the minimum cutting tool diameter of 6 mm to ensure machinability of the design. The actual $z_{step}$ value also depends on the step depth of the locating jaw to ensure both steps meet the supporting surfaces of the WP while the jaws are aligned in the x-y plane, as required by the fixture.

### Constraints on step-rest-jaw parameters

For the StR jaws the rest can be regarded as a box, despite the cutout at the contact surface. This facilitates the definition of the constraints. The minimum width of the rest in y-direction ($y_{rest}$ in Figure 5-4 c/d)) is required to be 10 mm, regardless of the fact whether the rest is on the left or the right side of the jaw. This is defined by requirement no.15. The maximum width is restricted by the *CoM* label. In y-direction, the center of mass of the WP has to be placed such that the WP does not tilt over the jaws under the influence of the gravity force when the vise is disengaged. This is required for stable (un-)loading (requirement no.5). The rest can have a different length ($x_{rest}$) than the step with the maximum being defined by $x_{step}$. The difference between the step length and the rest length also influences the minimum length of the step of the slide jaw. This is necessary to ensure that the Ø6 mm cutting tool can mill the steps. The height of the rest ($z_{rest}$) can also be different to the height of the step ($z_{step}$). The minimum height is defined by the 2 mm surface height requirement and the maximum is defined by the height of the step. A cutout at the inside corner of the rest is required to ensure the machinability of the design. Depending on the length of the rest, this cutout can be either made in x-direction or in y-direction. Further, the contact surface of the rest should be rounded to ensure a perpendicular contact when WP with plane but non-perpendicular primary and secondary locating surfaces are located. The rounding of the contact surface of the jaw results in a line contact that can always have a perpendicular contact normal to a plane surface. The radius of the rounded contact (r in Figure 5-4 c/d)) is user-defined to not be less than 0,5mm to ensure stability of the rest. The radii of the inside corner fillets are defined by the smallest tool diameter and are hence 3mm.

### Constraints on V-jaw parameters

For the V-jaw, it is defined that in front of the V-block a certain step length must remain to allow for the run-out of the cutting tool when cutting the V-pocket and hence to ensure the machinability of the V-pocket. The length is depending on the step length at the slide jaw as the sum of both lengths must be at least the minimum tool diameter of 6mm. The minimum length of the wedges ($x_w$) is set to 2 mm. This value is user-defined based on tests with the

smallest allowed WP radius which is 5 mm. The maximum length of the wedge is depending on the step length and the x-dimension of the remaining volume for machining shown in Figure 5-5. Thus, the maximum value can be 12 mm. The range of the width of the wedges ($y_w$) is defined by the required opening angle of the V-pocket. To achieve an opening angle (β) between 120° and 140° the angle of the wedges (α) has to be between 20° and 30°. As α is a function of the length and the width of the wedges, defining one parameter with a range and putting the other one in relation allows for a restriction of the opening angle range. A calculation of the range values leads to a minimum wedge width of 12/7* $x_w$, resulting in an opening angle of 119.5°, and a maximum wedge width of 8/3* $x_w$, resulting in an opening angle of 138.9°. As the opening angle requirement is flexible the slight alterations in the angle values are feasible. A further design variable is the placement of the wedges in y-direction ($y_{off}$ in Figure 5-4 b)). To meet requirement no.15, at least 10mm of material must remain between the outsides of the wedges and the outside of the jaw. Just like the height of the rest for the StR jaws, the height of the wedges ($z_w$) can be independent from the height of the step $z_{step}$. The minimum is, therefore, 2 mm and the maximum equals $z_{step}$. The V-shaped jaw design allows for a gap between the wedges ($y_{gap}$). The minimum value of that gap is 0 mm, meaning that the tips of the wedges touch. However, to ensure that the cylindrical WP surface touches only the sloped surfaces of the wedges, i.e. that requirement no.1 is met, the minimum can be higher. This is shown in Figure 10-1. The y-position of the contact point at the WP surface that leads to a contact normal that is perpendicular to the sloped wedge surface ($y_{ra}$) is defined by the wedge angle α. The outside edge of the wedge must at least be at this y-position to establish a contact at a right angle. If the width of the wedge ($y_w$) is smaller than $y_{ra}$, the minimum value of the gap ($y_{gap}$) must therefore be set such that $y_{gap}/2 + y_w = y_{ra}$. The maximum value of the gap also has to be constrained to ensure the jaw-WP contact is established between the sloped surfaces of the wedges and the cylindrical WP surface. Looking at Figure 10-1, it can be seen that when the wedges are moved apart, the value of $x_{gap}$ decreases. When $x_{gap}$ equals zero, the contact between the WP and the jaw is no longer established at the wedge surfaces only but also at the base surface, leading to an over-determined locating. When the wedges are moved apart even further, only the line contact at the base surface remains. To circumvent this issue a minimum $x_{gap}$ value of 0,5 mm is defined in the grammar.

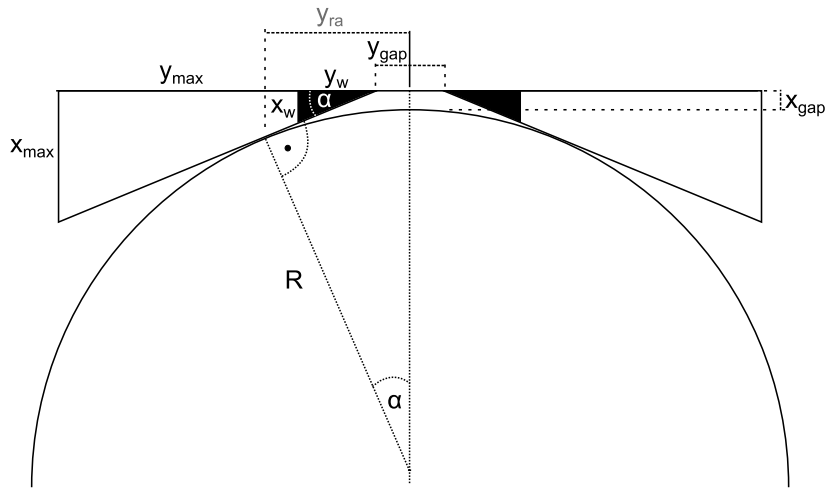



**Figure 10-1 Cylindrical workpiece surface located at two wedges building a V-shape including design parameters**

### Constraints on the WP geometry

The presented requirements and jaw design constraints impose some further constraints on the processible WP geometry. The minimum height of 2mm of the clamping and supporting surfaces of the jaws must also be respected when the cutting depths of the anvil and the slide jaw are different due to different WP supporting surfaces. Taking the maximum cutting depth of 22 mm into consideration, the two supporting surfaces of the WP can only be 20 mm apart from each other in z-direction. The maximum WP width (y-dimension) is limited to 140 mm by the minimum width of the rest ($y_{rest,min}$=10 mm) and the machining envelope of 150 mm in y-direction. For cylindrical WP surfaces, the maximum WP radius can be half the y-dimension of the machining envelope when the WP is located centered in the vise. The resulting maximum radius is 100 mm. The maximum clamping length restriction must still be maintained, meaning that the maximum diameter of a fully cylindrical WP is 126 mm. This value results from the 120 mm opening length of the vise plus the minimum step length at the anvil and slide jaw.

## 10.6.3 Details on rules and design rationale

### Rules R00-X

All R0-X rules match a set of labels on the LHS and generate two L-shaped bodies and either a box or two wedges on the RHS. The *CoM*, *Loc_1*, *Clamp*, *Support_1* and *Support_2* labels are present in all IWS. The presence, naming and positioning of the other labels determine which of the R0-X rules can be applied to an IWS. The placement parameters for most labels are unrestricted but some have a predefined range, e.g. to restrict the minimum and maximum distance of the label marking the primary locating surface and the label marking the clamping surface. This way, the opening length constraints (requirement no.10&11) are embedded in the

grammar. Further, the z-distance of the two support labels is restricted according to the machinable volume ($z_{step,min}$ and $z_{step,max}$ in Figure 5-5).

The L-shaped bodies on the RHS are built from 1 mm thick boxes and model the step-shaped cutout that is the basis for each jaw shape. The x-placement of the vertical boxes is defined by the *Loc_1* and *Clamp* labels. The z-placement of the horizontal boxes is defined by the *Support_1* and *Support_2* labels. The top surfaces of both L-shapes have to be in the same plane and, thus, the height of the L-shapes has to be variable to allow for different z-positions of the *Support_1* and *Support_2* labels. The congruency of the surfaces is necessary as they represent the top surfaces of the final jaws, which have to align when the jaws are placed in the fixture device. The width of the L-shapes is fixed to the width of the jaw blanks (requirement no.9). When applied, each R0-X rule deletes all but the *Loc_1* labels that are present in the IWS and adds a different set of labels instead. Most of these labels are only necessary for the assembly model creation that is part of the interference analysis, as explained in Section 6.1. Two labels *Support* are added instead of *Support_1* and *Support_2* as state control labels for subsequent rules. The points mentioned so far are the same for all versions of R0-X. The differences in the rules are as follows:

The rules R00-1 and R00-2 build a combination of step-rest-shaped jaw as locating jaw and a step-shaped jaw as clamping jaw when the *Loc_2* label is present. R00-1 applies when the *Loc_2* label has a yaw value between -60° and +60. The yaw value (rotation around z-axis) is defined by the direction of the outward facing normal of the secondary locating surface of the WP. A yaw value in the mentioned range represents a secondary locating surface that faces in positive y-direction. Hence, the respective contact surface of the jaw needs to face in the negative y-direction. In contrary, R00-2 applies when the *Loc_2* label has a yaw value between 120° and 240°. This resembles the case of a secondary locating surface facing in the negative y-direction. The y-placement of the L-shapes is variable within a range defined by the y-position of the *CoM* label and the y-position of the *Loc_2* label. Through this range restriction the requirement for a minimum remaining material thickness in y-direction and for stable (un-)loading of the WP are embedded in the grammar (requirement no.5&16). The y-placement for both L-shapes always has to be the same to ensure that the jaws are aligned. Additionally to the L-shapes, these two rules build a box (Rest) at the locating jaw that serves as rest in y-direction. This box is placed such that it aligns with one side of the jaw. Its width is generally the y-distance from the side of the jaw to the secondary WP locating surface and is, therefore, defined by the y-placement of the L-shapes, the *Loc_1* and the *Loc_2* label. This allows the rule to deal with WPs where the secondary WP locating surface meets the primary one at an acute, right or obtuse angle. The length and height of the rest box are variable within a range and define the minimum dimensional values of the L-shape at the locating side. The minimum and maximum values are defined by the machinable volume and the machinability requirements.

The R00-3 rule builds a combination of a V-shaped jaw as anvil jaw and a step-shaped jaw as slide jaw when a label *Loc_2_cyl* placed in positive y-direction from the *Loc_1* label is present in the IWS. Additionally to the L-shapes, the rule adds two wedges, Wedge0 and Wedge1, which build the V-block. The wedges are identical and only differ in their orientation and placement. The dimensions of the wedges and the gap between them are variable within ranges defined by the radius of the cylindrical WP surface (via the y-distance of *Loc_1* and *Loc_2_cyl*

label). These range definitions incorporate the requirement for accurate locating and the constraint of the opening angle of the V-block (requirements no.1&13). The y-placement of the *Loc_1* label defines the centerline for the (symmetrical) positioning of the wedges in the y-direction. The x-placement of the wedges depends on the x-placement of the *Loc_1* label, the WP surface radius and the y-placement of the wedges. The z-placement of the wedges is defined by the *Support_1* label. The placement of the L-shapes in y-direction is variable and depends on the y-placement of the wedges. As in the previous R00-X cases, the difference between the length of the horizontal box and the length of the wedges influence the minimum length of the horizontal box on the slide side, to ensure that the Ø6mm cutting tool can be used to machine the jaw design.

The last R00 rule, R00-4, generates a combination of two V-shaped jaws as anvil and slide jaw. This rule only applies when a *Loc_2_cyl* and a *Clamp_cyl* label are present in the IWS, meaning that both the locating and clamping surfaces of the WP have to be cylindrical. This rule builds four wedges, two on each jaw side. All variables and dependencies at the anvil jaw primitives equal the ones in the R00-V&St rule. The dimensions of the wedges on the slide jaw side are independent from the ones at the anvil jaw side. Their x-placement depend on the x-placement of the *Clamp* label and the radius of the WP clamping surface, defined by the y-placement of the *Clamp_cyl* label. The y-placement of the centerline of the V-cutout at the slide side has to be the same as on the anvil side. Due to this interdependency, the width of the wedges and the width of the gap between the wedges of both jaws have to be taken into account in the definition of the y-displacement of the centerline. Else the wedges at one of the jaws could extend beyond the y-boarders of the jaw. The z-placement is defined by the *Support_2* label. An IWS with cylindrical locating and clamping surfaces differs from an IWS with cylindrical locating and plane clamping surface through the presence of the *Clamp_cyl* label. Thus, an IWS that has the *Clamp* and *Clamp_cyl* label leads to a LHS match for the R00_V&St or the R00_V&V rule. By applying the rules in a random order both jaw set designs can therefore result in case of a cylindrical clamping surface.

### Rule R01

The state of the IWS only allows for the application of the R00 rules. All subsequent rules require the successful, collision-free application of one of the R00 rules to change the state of the *Support* labels. R01 becomes applicable once any of the R00 rules is applied. The rule matches the state label *Support* (in yellow) and an L-shape created from two thin boxes in close proximity to the *Support* label on the LHS. When applied, it extends the L-shape to create a step-shaped jaw with a total height of 40mm and a total length of 28mm and changes the color of the label to green. The height and length values comply with the dimensions of the jaw blanks. By allowing for slight variations in the placement of the boxes in regard to the *Support* label and the orientation of the boxes the rule can be used for both the anvil and the slide side. It can thus be applied twice to any current working shape as two yellow *Support* labels and two L-shapes exist when any R00 rule was applied.

### Rule R02

Rule R02 is used to create add-on boxes on the outside of existing wedges. The added boxes have the same height and length as the wedges and extend from the outside of the wedge to the respective outside of the jaw. Though these add-ons are not necessary for the function of the jaws they increase the structural stability of the design and reduce the volume that has to be machined. To be applicable, the rule requires the presence of a box, two wedges and a green *Support* label. Thus, it is only applicable after R01 was applied and only if a V-shaped jaw is present. The dimensions of the primitives on the LHS can vary but to make sure only a correct set of primitives is matched, their placement in regard to the *Support* label and in regard to each other is restricted. For example, the wedges have to be within a certain x-placement range to the *Support* label and they have to sit on top of the box. Just like in R01, the parameter ranges are defined in a way that allows for a matching of the rule on both the anvil and the slide side. This way, the rule can be either applied once, in case of a V&St set, or twice, in case of a V&Vset. When applied, the rule also changes the color of the *Support* label to blue.

### Rules R03-X

The application of rule R02 and the caused change of the *Support* label color to blue is the prerequisite for the application of rule R03. Two versions of this rule exist, R03_1 and R03_2. They both create a fillet of the inside corner(s) created by the tips of the wedges. This is necessary to ensure the machinability of the V-cutouts. A differentiation of the rule is required as two different approaches to model the fillets are necessary, depending on the size of the gap between the wedges. If the gap is small enough for the Ø6mm cutting tool to touch both sloped faces of the wedges simultaneously with the cylindrical cutting tool surface being tangent to the sloped wedge surfaces rule R03_2 can be applied. When a certain width of the gap is exceeded, depending on the angle of the wedges and the fixed cutting tool diameter, the tangency condition is no longer possible. Then rule R03_1 can be applied. Rule R03_2 adds one solid, rule R03_1 two, created from the Boolean intersection of a box and a cylinder. The cylinder represents the cutting tool and the box is necessary to fill up any existing gaps. For rule R03_1 the x- and the y-placement are defined by equations such that the tangency criterion between the cylindrical surface and the wedges holds. For rule R03_2 the cylinder is not moved in y-direction but placed at the center line of the wedges. Depending on the number of existing V-shaped jaws in the set, the R03 rules can be applied once or twice. When applied, the color of the *Support* label is changed to pink. This terminates the rule application process as no rules within the rule set match a *Support* label of that color.

### Rules R04-X

To ensure the machinability of any step-rest-shaped jaw generated by the grammar the four rules R04_1 to R04_4 are required. The rules create a cutout to make the corner of the StR jaw design machinable and round off the remaining secondary contact surface of the jaw. The LHS matching of all four rules requires two boxes touching each other in x-direction and aligned in y-direction, either on the left side or on the right side of the jaw. Further, the match of a green *Support* label, a yellow *Loc_2* label, an orange *CoM* label and a label named *Cutout*, which is created when one of the two R00-StR&St rules is applied, is required. The green *Support* label

restricts the applicability of the rules to the state after R01 is applied. If the length of Box1 is smaller than 7mm the cutout will affect both Box0 and Box1 to ensure that the remaining length of the y-contact element is at least 1mm (6 mm cutting tool diameter plus 1 mm rest length equals 7 mm). The cutout is based on a Boolean intersection of Box0 and Box1 with a body build from several Boolean operations. To ensure machinability of the cutout with 2.5D machining it always has to extend from the supporting surface level to the top of the jaw. The radii of the cutout are based on the minimum cutting tool radius.

By rounding off the jaw contact surface in y-direction a line contact is established between the WP and the jaw, which allows for the accurate positioning of WP with obtuse, right-angled or sharp-angled primary and secondary locating surface. The radius of the rounding is defined by the length of the rest box. If the length exceeds 7mm the cutout is always placed such that it is tangent to the front surface of the box that holds the functional features. Therefore, if the length of the rest box increases the radius of the secondary contact also increases. If the length, however, is less than 7mm the minimum radius of 0,5mm is used. The width of the cutout, defined by the y-placement, depends on the position and the length of the rest box. The longer the rest box is, the deeper the cutout needs to be to allow for a tangent transition between the cylindrical cutout and the rounding of the contact surface.

R04_1 and R04_2 apply to StR jaws with a negative y-locating direction while R04_3 and R04-4 apply to StR jaws with positive y-locating direction. R04_1 and R04_3 apply to WP with obtuse angles and R04_2 and R04_4 to WP with right or sharp angles. The differentiation of the y-locating direction is done with the yaw value of the *CoM* label, which is 180° for WP that require a negative y-locating element and 0° for ones that require a positive y-locating element. The differentiation between obtuse and right or sharp angles uses the yaw value of the *Loc_2* label, which always represents the outward normal direction of the secondary WP locating surface. Both labels are generated and oriented as part of the WP preparation process.

## 10.7 Spatial grammar evaluation data

| Setup ID | Jaw set design | Avg. no. of applications[6] | Max. no. of applications | Standard deviation | Ratio V&St/ V&V |
|---|---|---|---|---|---|
| Angled_Socket_0 | StR_Neg&St | 14,7 | 60 | 15,83 | -- |
| Angled_Socket_1 | StR_Neg&St | 4,7 | 6 | 0,67 | -- |
| AS_Acis09_0 | StR_Neg&St | 12,9 | 36 | 11,03 | -- |
| AS_Acis09_1 | V&St | 5,0 | 5 | 0 | -- |
| Cap_Arm_0 | StR_Pos&St | 10,4 | 17 | 4,78 | -- |
| Cap_Arm_1 | V&St/V&V | 6,1 | 8 | 1,58 | 6/4 |
| Cube_CS_75x40x30 _0 | StR_Neg&St | 4 | 4 | 0 | -- |
| Cube_CS_75x40x30 _1 | StR_Neg&St | 92,67 | 196 | 56,80 | -- |
| Cyl_D40_H70_0 | V&St/V&V | 6,2 | 7 | 0,98 | 5/5 |
| Cyl_D40_H70_1 | V&St/V&V | 6,1 | 8 | 1,04 | 7/3 |
| Cyl_D60_H15_0 | V&St/V&V | 12,7 | 24 | 6,36 | 9/1 |
| Cyl_Plane_0 | StR_Pos&St | 12,5 | 35 | 9,14 | -- |
| Cyl_Plane_1 | V&St | 6,5 | 11 | 1,80 | -- |
| Half_Butterfly_0 | StR_Pos&St | 12,1 | 40 | 10,99 | -- |
| Half_Butterfly_1 | V&St/V&V | 9,2 | 14 | 3,46 | 5/5 |
| Holder_0 | StR_Pos&St | 11,2 | 20 | 5,19 | -- |
| Holder_1 | StR_Neg&St | 4 | 4 | 4 | -- |
| Lever_0 | StR_Neg&St | 9,2 | 30 | 8,32 | -- |
| Lever_1 | V&St | 6,0 | 9 | 1,61 | -- |
| Pentagon_0 | V&St/V&V | 32,9 | 79 | 20,93 | 9/1 |
| Pentagon_1 | StR_Pos&St | 4 | 4 | 0 | -- |
| Rec_FR_0 | StR_Neg&St | 8,1 | 13 | 3,39 | -- |
| Rec_FR_1 | V&St/V&V | 65,1 | 135 | 36,53 | 5/5 |
| Winged_Flange_0 | StR_Pos&St | 4,8 | 7 | 0,87 | -- |
| Winged_Flange_1 | StR_Pos&St | 5,0 | 7 | 1,0 | -- |
| Winged_Flange_2 | StR_Neg&St | 17,2 | 57 | 14,97 | -- |

[6] Average no. of rule applications required to generate a solution with the jaw set design grammar; average taken from the one-time generation of ten solutions

## 10.8 NC codes for case-study setups interference analysis

### 10.8.1 Winged_Flange_Setup3of3

```
;===   cPost Standard PP for SINUMERIK 840 D        ===
;=====================================================
N10 G0 G90 G40
N20 G17
N30 ;=============== TOOL CHANGE =================
N40 ; DESC : T2 Schaftfräser D 10
N50 ;=============================================
N60 T2 M06
N70 D2
N80 G0 G90 G40 G17
N90 G94 F318 S3183 M3
N100 G64 SOFT
N110 G1 X63 Y-25 Z-14.667 F300 G94
N120 Z-7.667
N130 X56
N140 X50 F318
N150 X42.942
N160 X20.8 Y-64.856
N170 G2 X12.058 Y-70 I-8.742 J4.856
N180 G1 X-12.058
N190 G2 X-20.8 Y-64.856 I0 J10
N200 G1 X-42.942 Y-25
N210 X-50
N220 X-56
N230 G0 Z-14.167
N240 G1 Z-3.333 F300
N250 X-50 F318
N260 X-42.942
N270 X-20.8 Y-64.856
N280 G3 X-12.058 Y-70 I8.742 J4.856
N290 G1 X12.058
N300 G3 X20.8 Y-64.856 I0 J10
N310 G1 X42.942 Y-25
N320 X50
N330 X56
N340 G0 Z-9.833
N350 G1 Z1 F300
N360 X50 F318
N370 X42.942
N380 X20.8 Y-64.856
N390 G2 X12.058 Y-70 I-8.742 J4.856
N400 G1 X-12.058
N410 G2 X-20.8 Y-64.856 I0 J10
N420 G1 X-42.942 Y-25
N430 X-50
N440 X-56
N450 Z-12 F1
N460 Z-15
N470 M5 M9 M30
```

## 10.8.2 CylPlane_Setup2of2

```
;=====================================================
;===   cPost Standard PP for SINUMERIK 840 D        ===
;=====================================================
N10 G0 G90 G40
N20 G17
N30 ;=============== TOOL CHANGE ================
N40 ; DESC : T1 Stirnfräser D40
N50 ;============================================
N60 T1 M06
N70 D1
N80 G0 G90 G40 G17
N90 G94 F80 S796 M3
N100 G64 SOFT
N110 G1 X64.5 Y-30 Z-24 F80 G94
N120 Z-22
N130 X99
N140 G2 X100 Y-29 I0 J1
N150 G1 Y29
N160 G2 X99 Y30 I-1 J0
N170 G1 X30
N180 G2 X0 Y0 I0 J-30
N190 G2 X30 Y-30 I30 J0
N200 G1 X64.5
N210 Y-20
N220 Y-10
N230 X80
N240 Y10
N250 X30
N260 G2 X20 Y0 I0 J-10
N270 G2 X30 Y-10 I10 J0
N280 G1 X64.5
N290 Z-24 F1000
N300 G17
N310 ;=============== TOOL CHANGE ================
N320 ; DESC : T9 Schaftfräser D 15 Schruppfräser
N330 ;============================================
N340 T9 M06
N350 D9
N360 G0 G90 G40 G17
N370 G94 F1910 S3183 M3
N380 G64 SOFT
N390 G1 X30 Y-8 Z-24 F300
N400 Z-22
N410 Z-17.5
N420 X78 F1910
N430 Y8
N440 X30
N450 G2 X22 Y0 I0 J-8
N460 G2 X30 Y-8 I8 J0
N470 G1 Y-4.25
N480 Y-0.5
N490 X70.5
```

```
N500 Y0.5
N510 X30
N520 G2 X29.5 Y0 I0 J-0.5
N530 G2 X30 Y-0.5 I0.5 J0
N540 G0 Z-24.25
N550 G0 Y-8
N560 G1 Z-13 F300
N570 X78 F1910
N580 Y8
N590 X30
N600 G2 X22 Y0 I0 J-8
N610 G2 X30 Y-8 I8 J0
N620 G1 Y-4.25
N630 Y-0.5
N640 X70.5
N650 Y0.5
N660 X30
N670 G2 X29.5 Y0 I0 J-0.5
N680 G2 X30 Y-0.5 I0.5 J0
N690 G0 Z-19.75
N700 G0 Y-8
N710 G1 Z-8.5 F300
N720 X78 F1910
N730 Y8
N740 X30
N750 G2 X22 Y0 I0 J-8
N760 G2 X30 Y-8 I8 J0
N770 G1 Y-4.25
N780 Y-0.5
N790 X70.5
N800 Y0.5
N810 X30
N820 G2 X29.5 Y0 I0 J-0.5
N830 G2 X30 Y-0.5 I0.5 J0
N840 G0 Z-15.25
N850 G0 Y-8
N860 G1 Z-4 F300
N870 X78 F1910
N880 Y8
N890 X30
N900 G2 X22 Y0 I0 J-8
N910 G2 X30 Y-8 I8 J0
N920 G1 Y-4.25
N930 Y-0.5
N940 X70.5
N950 Y0.5
N960 X30
N970 G2 X29.5 Y0 I0 J-0.5
N980 G2 X30 Y-0.5 I0.5 J0
N990 G1 Z-22 F1000
N1000 Z-24
N1010 M5 M9
N1020 M30
```

## 10.8.3 Cyl_D60_H15_Setup1of1

```
;=====================================================
;===   cPost Standard PP for SINUMERIK 840 D        ===
;=====================================================
N10 G0 G90 G40
N20 ;=============== TOOL CHANGE ================
N30 ; DESC : T6 Schaftfräser D 10 Schruppfräser
N40 ;============================================
N50 T6 M06
N60 D6
N70 G0 G90 G40 G17
N80 G94 F2865 S4775 M3
N90 G64 SOFT
N100 G1 X968.407 Y1017.259 Z1000 F2865 G94
N110 X973.673 Y1014.383
N120 G2 X970 Y1000 I26.327 J-14.383
N130 G2 X1000 Y970 I30 J0
N140 G2 X1030 Y1000 I0 J30
N150 G2 X1000 Y1030 I-30 J0
N160 G2 X973.673 Y1014.383 I0 J-30
N170 G1 X975.866 Y1013.184
N180 X978.06 Y1011.986
N190 G2 X975 Y1000 I21.94 J-11.986
N200 G2 X1000 Y975 I25 J0
N210 G2 X1025 Y1000 I0 J25
N220 G2 X1000 Y1025 I-25 J0
N230 G2 X978.06 Y1011.986 I0 J-25
N240 G1 X980.254 Y1010.787
N250 X982.448 Y1009.589
N260 G2 X980 Y1000 I17.552 J-9.589
N270 G2 X1000 Y980 I20 J0
N280 G2 X1020 Y1000 I0 J20
N290 G2 X1000 Y1020 I-20 J0
N300 G2 X982.448 Y1009.589 I0 J-20
N310 G1 X984.642 Y1008.39
N320 X986.836 Y1007.191
N330 G2 X985 Y1000 I13.164 J-7.191
N340 G2 X1000 Y985 I15 J0
N350 G2 X1015 Y1000 I0 J15
N360 G2 X1000 Y1015 I-15 J0
N370 G2 X986.836 Y1007.191 I0 J-15
N380 G1 X989.03 Y1005.993
N390 X991.224 Y1004.794
N400 G2 X990 Y1000 I8.776 J-4.794
N410 G2 X1000 Y990 I10 J0
N420 G2 X1010 Y1000 I0 J10
N430 G2 X1000 Y1010 I-10 J0
N440 G2 X991.224 Y1004.794 I0 J-10
N450 G1 X993.418 Y1003.596
N460 X995.612 Y1002.397
N470 G2 X995 Y1000 I4.388 J-2.397
N480 G2 X1000 Y995 I5 J0
N490 G2 X1005 Y1000 I0 J5
```

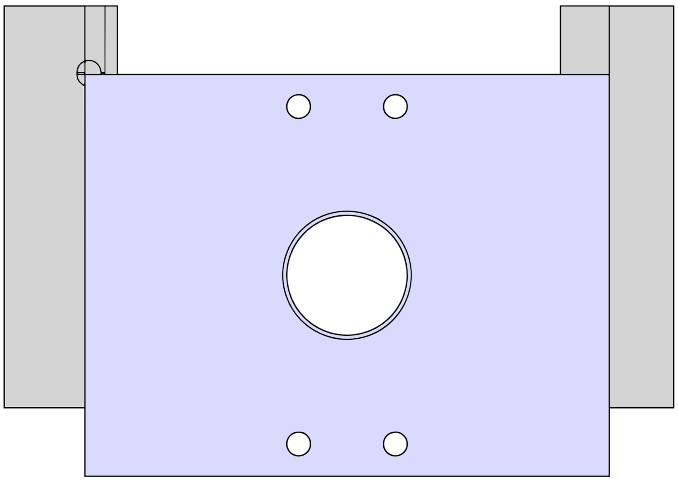
```
N500 G2 X1000 Y1005 I-5 J0
N510 G2 X995.612 Y1002.397 I0 J-5
N520 G1 Z998 F1
N530 X999.786 Y998.012 F300
N540 Z1003
N550 G3 X998 Y1000 I0.214 J1.988 F2865
N560 G3 X1000 Y1002 I2 J0
N570 G3 X1002 Y1000 I0 J-2
N580 G3 X1000 Y998 I-2 J0
N590 G3 X999.786 Y998.012 I0 J2
N600 G1 X999.518 Y995.526
N610 X999.25 Y993.04
N620 G3 X993 Y1000 I0.75 J6.96
N630 G3 X1000 Y1007 I7 J0
N640 G3 X1007 Y1000 I0 J-7
N650 G3 X1000 Y993 I-7 J0
N660 G3 X999.25 Y993.04 I0 J7
N670 G1 X998.983 Y990.555
N680 X998.715 Y988.069
N690 G3 X988 Y1000 I1.285 J11.931
N700 G3 X1000 Y1012 I12 J0
N710 G3 X1012 Y1000 I0 J-12
N720 G3 X1000 Y988 I-12 J0
N730 G3 X998.715 Y988.069 I0 J12
N740 G1 X998.447 Y985.583
N750 X998.179 Y983.098
N760 G3 X983 Y1000 I1.821 J16.902
N770 G3 X1000 Y1017 I17 J0
N780 G3 X1017 Y1000 I0 J-17
N790 G3 X1000 Y983 I-17 J0
N800 G3 X998.179 Y983.098 I0 J17
N810 G1 X997.912 Y980.612
N820 X997.644 Y978.127
N830 G3 X978 Y1000 I2.356 J21.873
N840 G3 X1000 Y1022 I22 J0
N850 G3 X1022 Y1000 I0 J-22
N860 G3 X1000 Y978 I-22 J0
N870 G3 X997.644 Y978.127 I0 J22
N880 G1 Z1000 F1
N890 Z998
N900 ;=============== TOOL CHANGE ================
N910 ; DESC : T4 Schaftfräser D 6
N920 ;============================================
N930 T4 M06
N940 D4
N950 G0 G90 G40 G17
N960 G94 F4775 S7958 M3
N970 G64 SOFT
N980 G1 X990.25 Y1016.887 Z999 F300
N990 X990.2 Y1016.974 Z1004
N1000 G2 X990.15 Y1016.887 I0.05 J-0.087 F4775
N1010 G2 X990.25 Y1016.787 I0.1 J0
N1020 G2 X990.3 Y1016.801 I0 J0.1
N1030 G2 X980.6 Y1000 I9.7 J-16.801
```

```
N1040 G2 X990.3 Y983.199 I19.4 J0
N1050 G3 X990.35 Y983.113 I-0.05 J-0.086
N1060 G3 X990.25 Y983.013 I-0.1 J0
N1070 G3 X990.2 Y983.026 I0 J0.1
N1080 G3 X980.4 Y1000 I9.8 J16.974
N1090 G3 X990.2 Y1016.974 I19.6 J0
N1100 G0 Z998
N1110 G1 Z1008 F300
N1120 G2 X990.15 Y1016.887 I0.05 J-0.087 F4775
N1130 G2 X990.25 Y1016.787 I0.1 J0
N1140 G2 X990.3 Y1016.801 I0 J0.1
N1150 G2 X980.6 Y1000 I9.7 J-16.801
N1160 G2 X990.3 Y983.199 I19.4 J0
N1170 G3 X990.35 Y983.113 I-0.05 J-0.086
N1180 G3 X990.25 Y983.013 I-0.1 J0
N1190 G3 X990.2 Y983.026 I0 J0.1
N1200 G3 X980.4 Y1000 I9.8 J16.974
N1210 G3 X990.2 Y1016.974 I19.6 J0
N1220 G0 Z1002
N1230 G1 Z1012 F300
N1240 G2 X990.15 Y1016.887 I0.05 J-0.087 F4775
N1250 G2 X990.25 Y1016.787 I0.1 J0
N1260 G2 X990.3 Y1016.801 I0 J0.1
N1270 G2 X980.6 Y1000 I9.7 J-16.801
N1280 G2 X990.3 Y983.199 I19.4 J0
N1290 G3 X990.35 Y983.113 I-0.05 J-0.086
N1300 G3 X990.25 Y983.013 I-0.1 J0
N1310 G3 X990.2 Y983.026 I0 J0.1
N1320 G3 X980.4 Y1000 I9.8 J16.974
N1330 G3 X990.2 Y1016.974 I19.6 J0
N1340 G0 Z1006
N1350 G1 Z1016 F300
N1360 G2 X990.15 Y1016.887 I0.05 J-0.087 F4775
N1370 G2 X990.25 Y1016.787 I0.1 J0
N1380 G2 X990.3 Y1016.801 I0 J0.1
N1390 G2 X980.6 Y1000 I9.7 J-16.801
N1400 G2 X990.3 Y983.199 I19.4 J0
N1410 G3 X990.35 Y983.113 I-0.05 J-0.086
N1420 G3 X990.25 Y983.013 I-0.1 J0
N1430 G3 X990.2 Y983.026 I0 J0.1
N1440 G3 X980.4 Y1000 I9.8 J16.974
N1450 G3 X990.2 Y1016.974 I19.6 J0
N1460 G1 Z1000 F1
N1470 Z998
N1480 X1009.75 Y1016.887 Z999 F300
N1490 X1009.8 Y1016.974 Z1004
N1500 G3 X1019.6 Y1000 I-9.8 J-16.974 F4775
N1510 G3 X1009.8 Y983.026 I-19.6 J0
N1520 G1 X1009.7 Y983.199
N1530 G2 X1019.4 Y1000 I-9.7 J16.801
N1540 G2 X1009.7 Y1016.801 I-19.4 J0
N1550 G0 Z998
N1560 G1 X1009.8 Y1016.974 Z1008 F300
N1570 G3 X1019.6 Y1000 I-9.8 J-16.974 F4775
```

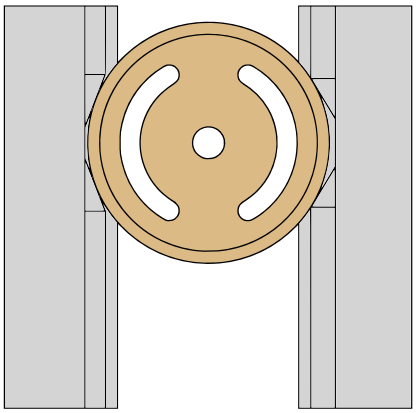
```
N1580 G3 X1009.8 Y983.026 I-19.6 J0
N1590 G1 X1009.7 Y983.199
N1600 G2 X1019.4 Y1000 I-9.7 J16.801
N1610 G2 X1009.7 Y1016.801 I-19.4 J0
N1620 G0 Z1002
N1630 G1 X1009.8 Y1016.974 Z1012 F300
N1640 G3 X1019.6 Y1000 I-9.8 J-16.974 F4775
N1650 G3 X1009.8 Y983.026 I-19.6 J0
N1660 G1 X1009.7 Y983.199
N1670 G2 X1019.4 Y1000 I-9.7 J16.801
N1680 G2 X1009.7 Y1016.801 I-19.4 J0
N1690 G0 Z1006
N1700 G1 X1009.8 Y1016.974 Z1016 F300
N1710 G3 X1019.6 Y1000 I-9.8 J-16.974 F4775
N1720 G3 X1009.8 Y983.026 I-19.6 J0
N1730 G1 X1009.7 Y983.199
N1740 G2 X1019.4 Y1000 I-9.7 J16.801
N1750 G2 X1009.7 Y1016.801 I-19.4 J0
N1760 G1 Z1000 F1
N1770 Z998
N1780 G0 X1000 Y1000 Z1002
N1790 G1 Z1006.25 F4775
N1800 X999.318 Y1000.731
N1810 G3 X1000 Y1001 I0.682 J-0.731
N1820 G3 X1000.731 Y1000.682 I0 J-1
N1830 G3 X1001 Y1000 I-0.731 J-0.682
N1840 G3 X1000 Y999 I-1 J0
N1850 G3 X999.062 Y999.654 I0 J1
N1860 G3 X999 Y1000 I0.938 J0.346
N1870 G3 X1000 Y1001 I1 J0
N1880 G3 X1001 Y1000 I0 J-1
N1890 G3 X1000.731 Y999.318 I-1 J0
N1900 G3 X1000 Y999 I-0.731 J0.682
N1910 G3 X999.318 Y999.269 I0 J1
N1920 G1 X1000 Y1000
N1930 Z1009.5
N1940 X999.318 Y1000.731
N1950 G3 X1000 Y1001 I0.682 J-0.731
N1960 G3 X1000.731 Y1000.682 I0 J-1
N1970 G3 X1001 Y1000 I-0.731 J-0.682
N1980 G3 X1000 Y999 I-1 J0
N1990 G3 X999.062 Y999.654 I0 J1
N2000 G3 X999 Y1000 I0.938 J0.346
N2010 G3 X1000 Y1001 I1 J0
N2020 G3 X1001 Y1000 I0 J-1
N2030 G3 X1000.731 Y999.318 I-1 J0
N2040 G3 X1000 Y999 I-0.731 J0.682
N2050 G3 X999.318 Y999.269 I0 J1
N2060 G1 X1000 Y1000
N2070 Z1012.75
N2080 X999.318 Y1000.731
N2090 G3 X1000 Y1001 I0.682 J-0.731
N2100 G3 X1000.731 Y1000.682 I0 J-1
N2110 G3 X1001 Y1000 I-0.731 J-0.682
```

```
N2120 G3 X1000 Y999 I-1 J0
N2130 G3 X999.062 Y999.654 I0 J1
N2140 G3 X999 Y1000 I0.938 J0.346
N2150 G3 X1000 Y1001 I1 J0
N2160 G3 X1001 Y1000 I0 J-1
N2170 G3 X1000.731 Y999.318 I-1 J0
N2180 G3 X1000 Y999 I-0.731 J0.682
N2190 G3 X999.318 Y999.269 I0 J1
N2200 G1 X1000 Y1000
N2210 Z1016
N2220 X999.318 Y1000.731
N2230 G3 X1000 Y1001 I0.682 J-0.731
N2240 G3 X1000.731 Y1000.682 I0 J-1
N2250 G3 X1001 Y1000 I-0.731 J-0.682
N2260 G3 X1000 Y999 I-1 J0
N2270 G3 X999.062 Y999.654 I0 J1
N2280 G3 X999 Y1000 I0.938 J0.346
N2290 G3 X1000 Y1001 I1 J0
N2300 G3 X1001 Y1000 I0 J-1
N2310 G3 X1000.731 Y999.318 I-1 J0
N2320 G3 X1000 Y999 I-0.731 J0.682
N2330 G3 X999.318 Y999.269 I0 J1
N2340 G1 X1000 Y1000
N2350 Z1002
N2360 Z999 F1
N2370 M5 M9
N2380 M30
```

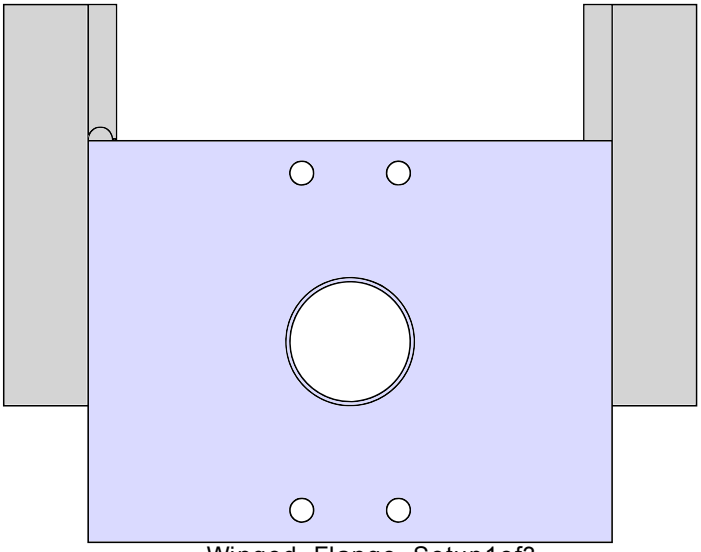
## 10.9 FEA - FD candidates used for evaluation of approach



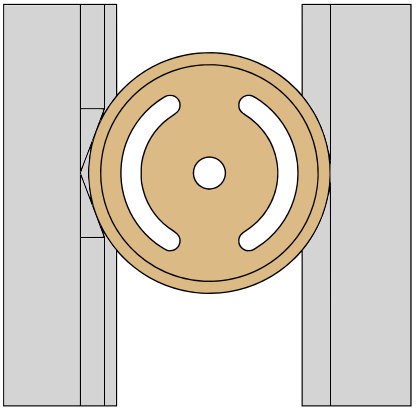
Winged_Flange_Setup1of3
Solution2



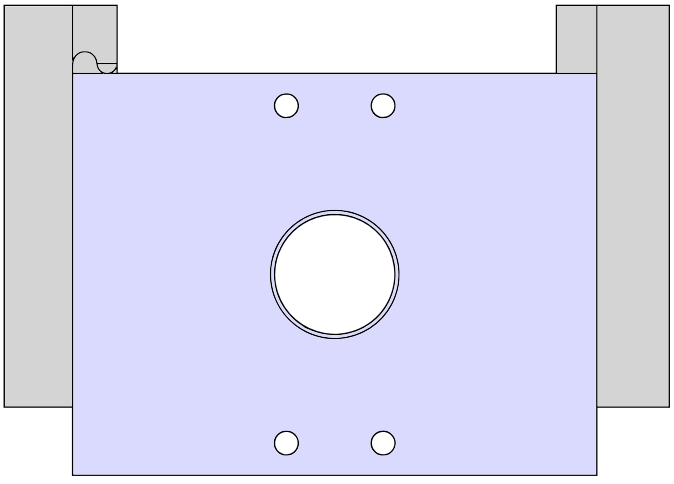
Cyl_D60_H15
Solution1



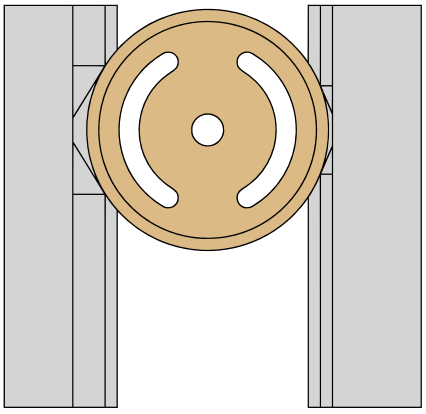
Winged_Flange_Setup1of3
Solution3



Cyl_D60_H15
Solution5



Winged_Flange_Setup1of3
Solution6



Cyl_D60_H15
Solution7

## 10.10 Data for machining and clamping force calculation

### Workpiece material data Al 6061 – T6 aluminum:

| Abbr. | Description | SI | Value | Source |
|---|---|---|---|---|
| 1-$m_c$ | Cutting force exponent | -- | 0,66 | For AlSi materials (PEROVIĆ 2013@38) |
| 1-$m_f$ | Feeding force exponent | -- | 0,13 | For AlSi materials (PEROVIĆ 2013@51) |
| $k_{c1.1}$ | Specific cutting force | N/mm² | 750 | For AlSi materials (PEROVIĆ 2013@38) |
| $k_{f1.1}$ | Specific feeding force | N/mm² | 90 | For AlSi materials (PEROVIĆ 2013@51) |

### Input values used for milling operations, tool T1:

| Abbr. | Description | SI | Value | Source |
|---|---|---|---|---|
| $a_p$ | Cutting depth | mm | 2 | Machining plan (NC file) |
| $a_c$=D | Cutting width | mm | 40 | Tool geometry/Worst case |
| $\kappa_r$ | Tool rake angle | Deg° | 90 | Tool geometry |
| z | No. of teeth | -- | 8 | Tool geometry |
| $\varphi_s$ | Tool enclosure angle | Deg° | 180 | Tool geometry/Worst case |
| n | Spindle speed | 1/min | 1450 | Machining plan (NC file) |
| f | Feed rate | mm/min | 800 | Machining plan (NC file) |
| K | Correction factor | -- | 2,0 | User defined (represents tool wear,…) |

### Input values used for milling operations, tool T2:

| Abbr. | Description | SI | Value | Source |
|---|---|---|---|---|
| $a_p$ | Cutting depth | mm | 3 | Machining plan (NC file) |
| $a_c$=D | Cutting width | mm | 10 | Tool geometry/Worst case |
| $\kappa_r$ | Tool rake angle | Deg° | 90 | Tool geometry |
| z | No. of teeth | -- | 3 | Tool geometry |
| $\varphi_s$ | Tool enclosure angle | Deg° | 180 | Tool geometry/Worst case |
| n | Spindle speed | 1/min | 4800 | Machining plan (NC file) |
| f | Feed rate | mm/min | 2900 | Machining plan (NC file) |
| K | Correction factor | -- | 2,0 | User defined (represents tool wear,…) |

### Input values used for milling operations, tool T4:

| Abbr. | Description | SI | Value | Source |
|---|---|---|---|---|
| $a_p$ | Cutting depth | mm | 3 | Machining plan (NC file) |
| $a_e$=D | Cutting width | mm | 6 | Tool geometry/Worst case |
| $\kappa_r$ | Tool rake angle | Deg° | 90 | Tool geometry |
| z | No. of teeth | -- | 2 | Tool geometry |
| $\varphi_s$ | Tool enclosure angle | Deg° | 180 | Tool geometry/Worst case |
| n | Spindle speed | 1/min | 8000 | Machining plan (NC file) |
| f | Feed rate | mm/min | 4800 | Machining plan (NC file) |
| K | Correction factor | -- | 2,0 | User defined (represents tool wear,…) |

### Input values used for drilling operations, tool T10:

| Abbr. | Description | SI | Value | Source |
|---|---|---|---|---|
| D | Outer diameter drill | mm | 6 | Tool geometry |
| $\kappa_r$ | Tool rake angle | Deg° | 60 | Tool geometry |
| z | No. of teeth | -- | 2 | Tool geometry |
| n | Spindle speed | 1/min | 8000 | Machining plan (NC file) |
| f | Feed rate | mm/min | 4800 | Machining plan (NC file) |
| K | Correction factor | -- | 1,2 | User defined (represents tool wear) |

## 10.11 FEA results

### 10.11.1 Comparative table for all studies

| Winged_Flange_Setup1of3_Candidate2 | Min DX | Max DX | Min DY | Max DY | Min DZ | Max DZ |
|---|---|---|---|---|---|---|
| Clamp | -1,14E-05 | -1,73E-06 | -6,44E-07 | 4,02E-06 | -5,86E-05 | 6,87E-07 |
| Clamp+Load_0 | -1,31E-05 | -3,86E-06 | -2,13E-06 | 1,65E-06 | -5,90E-05 | 7,44E-07 |
| Clamp+Load_1 | -6,83E-05 | -2,59E-06 | -7,32E-05 | -7,15E-06 | **-1,68E-04** | 3,34E-06 |
| Clamp+Load_2 | -1,47E-05 | -2,44E-06 | -4,55E-07 | 4,79E-06 | -5,40E-05 | 1,87E-06 |
| Clamp+Load_3 | -1,33E-05 | -2,92E-06 | -4,69E-07 | 4,94E-06 | -5,23E-05 | 1,12E-06 |
| Clamp+Load_4 | -1,54E-05 | -1,79E-06 | -7,46E-07 | 3,66E-06 | -4,96E-05 | 4,13E-07 |
| Clamp+Load_5 | -1,46E-05 | -1,99E-06 | -5,85E-07 | 3,51E-06 | -4,65E-05 | 3,80E-07 |
| **Winged_Flange_Setup1of3_Candidate3** | **Min DX** | **Max DX** | **Min DY** | **Max DY** | **Min DZ** | **Max DZ** |
| Clamp | -2,46E-05 | -6,74E-06 | -5,81E-07 | 9,10E-06 | -1,17E-04 | 2,11E-06 |
| Clamp+Load_0 | -2,87E-05 | -8,78E-06 | -3,34E-06 | 5,78E-06 | -1,20E-04 | 2,62E-06 |
| Clamp+Load_1 | -9,79E-05 | -1,92E-05 | -8,46E-05 | -1,50E-05 | **-3,19E-04** | 8,77E-06 |
| Clamp+Load_2 | -2,54E-05 | -6,47E-06 | -1,92E-07 | 1,07E-05 | -1,10E-04 | 6,60E-06 |
| Clamp+Load_3 | -2,21E-05 | -6,62E-06 | -2,47E-07 | 1,10E-05 | -1,08E-04 | 3,71E-06 |
| Clamp+Load_4 | -2,47E-05 | -7,18E-06 | -6,46E-07 | 8,51E-06 | -1,04E-04 | 1,78E-06 |
| Clamp+Load_5 | -2,25E-05 | -7,80E-06 | -5,56E-07 | 8,18E-06 | -9,92E-05 | 1,74E-06 |
| **Winged_Flange_Setup1of3_Candidate6** | **Min DX** | **Max DX** | **Min DY** | **Max DY** | **Min DZ** | **Max DZ** |
| Clamp | -8,91E-06 | -1,63E-06 | -5,61E-07 | 3,68E-06 | -4,44E-05 | 1,35E-06 |
| Clamp+Load_0 | -1,00E-05 | -2,60E-06 | -1,86E-06 | 1,50E-06 | -4,40E-05 | 3,08E-07 |
| Clamp+Load_1 | -5,90E-05 | 9,02E-08 | -6,89E-05 | -6,02E-06 | **-1,32E-04** | -8,46E-08 |
| Clamp+Load_2 | -1,19E-05 | -1,90E-06 | -4,37E-07 | 4,41E-06 | -4,08E-05 | 2,04E-06 |
| Clamp+Load_3 | -1,18E-05 | -1,86E-06 | -4,33E-07 | 4,88E-06 | -3,89E-05 | 1,45E-06 |
| Clamp+Load_4 | -1,29E-05 | -1,55E-06 | -9,11E-07 | 3,36E-06 | -3,71E-05 | 9,36E-07 |
| Clamp+Load_5 | -1,32E-05 | -1,65E-06 | -4,85E-07 | 3,22E-06 | -3,40E-05 | 8,67E-07 |

| Cyl_D60_H15_Setup1of1_Candidate1 | Min DX | Max DX | Min DY | Max DY | Min DZ | Max DZ |
|---|---|---|---|---|---|---|
| Clamp | -3,89E-05 | -1,03E-05 | -2,32E-05 | 1,90E-06 | -7,41E-05 | 1,58E-06 |
| Clamp+Load_0 | -3,29E-04 | -1,05E-04 | -2,74E-04 | -8,30E-05 | **-5,67E-04** | 4,27E-05 |
| Clamp+Load_1 | -5,05E-05 | -1,15E-05 | -2,80E-05 | 4,04E-07 | -8,96E-05 | 1,73E-06 |
| Clamp+Load_2 | -4,86E-05 | -1,16E-05 | -3,00E-05 | 9,67E-07 | -8,79E-05 | 1,77E-06 |
| Clamp+Load_3 | -4,34E-05 | -7,50E-06 | -2,39E-05 | 1,04E-06 | -5,04E-05 | 1,23E-06 |
| **Cyl_D60_H15_Setup1of1_Candidate5** | **Min DX** | **Max DX** | **Min DY** | **Max DY** | **Min DZ** | **Max DZ** |
| Clamp | -6,20E-05 | -2,04E-05 | -1,80E-05 | 1,56E-06 | -1,18E-04 | 5,89E-06 |
| Clamp+Load_0 | -5,93E-04 | -1,48E-04 | -3,51E-04 | -1,29E-04 | **-8,94E-04** | 9,78E-05 |
| Clamp+Load_1 | -7,78E-05 | -2,51E-05 | -2,53E-05 | -7,13E-07 | -1,37E-04 | 6,71E-06 |
| Clamp+Load_2 | -7,47E-05 | -2,64E-05 | -2,77E-05 | -3,07E-07 | -1,35E-04 | 6,63E-06 |
| Clamp+Load_3 | -5,89E-05 | -2,31E-05 | -1,77E-05 | 1,27E-06 | -8,59E-05 | 4,66E-06 |
| **Cyl_D60_H15_Setup1of1_Candidate7** | **Min DX** | **Max DX** | **Min DY** | **Max DY** | **Min DZ** | **Max DZ** |
| Clamp | -5,05E-05 | -1,46E-05 | -4,22E-05 | 1,27E-06 | -1,00E-04 | 3,72E-06 |
| Clamp+Load_0 | -3,87E-04 | -1,03E-04 | -3,19E-04 | -1,00E-04 | **-6,86E-04** | 7,29E-05 |
| Clamp+Load_1 | -6,15E-05 | -1,87E-05 | -4,89E-05 | -8,58E-07 | -1,16E-04 | 4,10E-06 |
| Clamp+Load_2 | -6,09E-05 | -2,04E-05 | -5,24E-05 | -4,30E-08 | -1,14E-04 | 4,12E-06 |
| Clamp+Load_3 | -4,72E-05 | -1,43E-05 | -4,30E-05 | 4,95E-07 | -7,66E-05 | 2,92E-06 |

## 10.11.2 Result files generated

### Winged_Flange_Setup1of3_Candidate2

```
          -- CODE_ASTER -- VERSION : EXPLOITATION (stable) --

               Version 11.5.0 modifiÃ©e le 10/12/2013
                rÃ©vision 2b68bca7 - branche 'v11'
                  Copyright EDF R&D 1991 - 2014

              ExÃ©cution du : Wed Jun 18 15:10:35 2014
              Nom de la machine : thomas-VBox-Ubuntu14
                      Architecture : 64bit
                  Type de processeur : x86_64
            SystÃ¨me d'exploitation : Linux 3.13.0-24-generic
                 Langue des messages : de (UTF-8)

                    Version de Python : 2.7.3
                     Version de NumPy : 1.7.1
                   ParallÃ©lisme MPI : inactif
                  ParallÃ©lisme OpenMP : actif
                 Nombre de processus utilisÃ©s : 1
               Version de la librairie HDF5 : 1.8.10
                Version de la librairie MED : 3.0.7
               Version de la librairie MUMPS : 4.10.0
              Version de la librairie SCOTCH : 5.1.10
             MÃ©moire limite pour l'exÃ©cution : 762.00 Mo
              consommÃ©e par l'initialisation : 213.68 Mo
             par les objets du jeu de commandes : 2.81 Mo
            reste pour l'allocation dynamique : 544.86 Mo
           Taille limite des fichiers d'Ã©change : 48.00 Go


  !--------------------------------------------------------------------------------!
  ! <A> <SUPERVIS_42>                                                              !
  !                                                                                !
  ! Les fichiers suivants ont Ã©tÃ© modifiÃ©s par rapport Ã  la derniÃ¨re rÃ©vision 2b68bca7 : !
  !                                                                                !
  ! checkout.log, configure.log, hg_diff.log, install.log, make.log                !
  !                                                                                !
  !                                                                                !
  ! Ceci est une alarme. Si vous ne comprenez pas le sens de cette                 !
  ! alarme, vous pouvez obtenir des rÃ©sultats inattendus !                         !
  !--------------------------------------------------------------------------------!


 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_0 CALCULE LE 18/06/2014 A 15:10:37 DE TYPE
EVOL_ELAS

                ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface
```

```
 ======>

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -1.73028747448504E-06 EN    1 NOEUD(S) : N48
 LA VALEUR MAXIMALE DE DY       EST  4.01817522060201E-06 EN    1 NOEUD(S) : N671
 LA VALEUR MAXIMALE DE DZ       EST  6.86576994269177E-07 EN    1 NOEUD(S) : N47
 LA VALEUR MINIMALE DE DX       EST -1.14130924716675E-05 EN    1 NOEUD(S) : N818
 LA VALEUR MINIMALE DE DY       EST -6.43614893404447E-07 EN    1 NOEUD(S) : N716
 LA VALEUR MINIMALE DE DZ       EST -5.86435165186829E-05 EN    1 NOEUD(S) : N743

 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_1 CALCULE LE 18/06/2014 A 15:10:42 DE TYPE
EVOL_ELAS

                ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface


 ======>

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -3.86484642030731E-06 EN    1 NOEUD(S) : N46
 LA VALEUR MAXIMALE DE DY       EST  1.64634330405410E-06 EN    1 NOEUD(S) : N671
 LA VALEUR MAXIMALE DE DZ       EST  7.44286115640837E-07 EN    1 NOEUD(S) : N45
 LA VALEUR MINIMALE DE DX       EST -1.31301853081947E-05 EN    1 NOEUD(S) : N818
 LA VALEUR MINIMALE DE DY       EST -2.13253752504629E-06 EN    1 NOEUD(S) : N48
 LA VALEUR MINIMALE DE DZ       EST -5.90396731904441E-05 EN    1 NOEUD(S) : N743

 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_2 CALCULE LE 18/06/2014 A 15:10:48 DE TYPE
EVOL_ELAS

                ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface


 ======>

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -2.59275851075356E-06 EN    1 NOEUD(S) : N46
 LA VALEUR MAXIMALE DE DY       EST -7.14926453611166E-06 EN    1 NOEUD(S) : N46
 LA VALEUR MAXIMALE DE DZ       EST  3.33881531511265E-06 EN    1 NOEUD(S) : N45
 LA VALEUR MINIMALE DE DX       EST -6.82786110013872E-05 EN    1 NOEUD(S) : N654
 LA VALEUR MINIMALE DE DY       EST -7.31869496179860E-05 EN    1 NOEUD(S) : N650
 LA VALEUR MINIMALE DE DZ       EST -1.67904302499555E-04 EN    1 NOEUD(S) : N807
```

```
 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_3 CALCULE LE 18/06/2014 A 15:10:53 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface


 ======>

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -2.44297660851084E-06 EN    1 NOEUD(S) : N859
 LA VALEUR MAXIMALE DE DY      EST  4.78989397233515E-06 EN    1 NOEUD(S) : N671
 LA VALEUR MAXIMALE DE DZ      EST  1.87244164095716E-06 EN    1 NOEUD(S) : N47
 LA VALEUR MINIMALE DE DX      EST -1.46593158519775E-05 EN    1 NOEUD(S) : N872
 LA VALEUR MINIMALE DE DY      EST -4.55468646317642E-07 EN    1 NOEUD(S) : N715
 LA VALEUR MINIMALE DE DZ      EST -5.40499468963800E-05 EN    1 NOEUD(S) : N743

 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_4 CALCULE LE 18/06/2014 A 15:10:58 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface


 ======>

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -2.91837157876598E-06 EN    1 NOEUD(S) : N859
 LA VALEUR MAXIMALE DE DY      EST  4.93569459837185E-06 EN    1 NOEUD(S) : N671
 LA VALEUR MAXIMALE DE DZ      EST  1.12098722195940E-06 EN    1 NOEUD(S) : N47
 LA VALEUR MINIMALE DE DX      EST -1.33358229146440E-05 EN    1 NOEUD(S) : N869
 LA VALEUR MINIMALE DE DY      EST -4.68851754000466E-07 EN    1 NOEUD(S) : N715
 LA VALEUR MINIMALE DE DZ      EST -5.22677726083556E-05 EN    1 NOEUD(S) : N743

 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_5 CALCULE LE 18/06/2014 A 15:11:03 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface


 ======>

 ------>
 Min/Max displacement of Active_Surface
```

```
 LA VALEUR MAXIMALE DE DX      EST -1.79218055168879E-06 EN    1 NOEUD(S) : N48
 LA VALEUR MAXIMALE DE DY      EST  3.65879655886410E-06 EN    1 NOEUD(S) : N671
 LA VALEUR MAXIMALE DE DZ      EST  4.12757886161216E-07 EN    1 NOEUD(S) : N45
 LA VALEUR MINIMALE DE DX      EST -1.54052342031358E-05 EN    1 NOEUD(S) : N884
 LA VALEUR MINIMALE DE DY      EST -7.45748978462884E-07 EN    1 NOEUD(S) : N884
 LA VALEUR MINIMALE DE DZ      EST -4.96014275416355E-05 EN    1 NOEUD(S) : N745

 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_6 CALCULE LE 18/06/2014 A 15:11:08 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface


 ======>

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.98566032965214E-06 EN    1 NOEUD(S) : N48
 LA VALEUR MAXIMALE DE DY      EST  3.51364073645059E-06 EN    1 NOEUD(S) : N671
 LA VALEUR MAXIMALE DE DZ      EST  3.80431475722834E-07 EN    1 NOEUD(S) : N45
 LA VALEUR MINIMALE DE DX      EST -1.46034350653773E-05 EN    1 NOEUD(S) : N878
 LA VALEUR MINIMALE DE DY      EST -5.84548447481652E-07 EN    1 NOEUD(S) : N716
 LA VALEUR MINIMALE DE DZ      EST -4.64771051392694E-05 EN    1 NOEUD(S) : N803
 <I> <FIN> FERMETURE DE LA BASE "GLOBALE" EFFECTUEE.

 <FIN> ArrÃªt normal dans "FIN".
 <I> <FIN> ARRET NORMAL DANS "FIN" PAR APPEL A "JEFINI".

 <I> <FIN> MEMOIRE JEVEUX MINIMALE REQUISE POUR L'EXECUTION :
60.04 Mo
 <I> <FIN> MEMOIRE JEVEUX OPTIMALE REQUISE POUR L'EXECUTION :
332.07 Mo
 <I> <FIN> MAXIMUM DE MEMOIRE UTILISEE PAR LE PROCESSUS LORS DE
L'EXECUTION :     732.54 Mo

 ********************************************************************************
 * COMMAND              :      USER :     SYSTEM :   USER+SYS :    ELAPSED *
 ********************************************************************************
 * init (jdc)           :      0.36 :      0.08 :      0.44 :      0.50 *
 *  . compile           :      0.00 :      0.00 :      0.00 :      0.00 *
 *  . exec_compile      :      0.13 :      0.04 :      0.17 :      0.20 *
 *  . report            :      0.03 :      0.00 :      0.03 :      0.05 *
 *  . build             :      0.00 :      0.00 :      0.00 :      0.00 *
 * DEBUT                :      0.04 :      0.04 :      0.08 :      0.15 *
 * DEFI_MATERIAU        :      0.01 :      0.00 :      0.01 :      0.02 *
 * DEFI_MATERIAU        :      0.00 :      0.00 :      0.00 :      0.00 *
 * LIRE_MAILLAGE        :      0.33 :      0.05 :      0.38 :      0.49 *
 * MODI_MAILLAGE        :      0.26 :      0.02 :      0.28 :      0.29 *
 * AFFE_MODELE          :      0.13 :      0.02 :      0.15 :      0.17 *
```

```
* AFFE_MATERIAU        :     0.01 :     0.01 :     0.02 :     0.02 *
* AFFE_CHAR_MECA       :     0.04 :     0.01 :     0.05 :     0.05 *
* AFFE_CHAR_MECA       :     0.03 :     0.01 :     0.04 :     0.04 *
* AFFE_CHAR_MECA       :     0.04 :     0.01 :     0.05 :     0.06 *
* AFFE_CHAR_MECA       :     0.05 :     0.02 :     0.07 :     0.07 *
* AFFE_CHAR_MECA       :     0.03 :     0.01 :     0.04 :     0.05 *
* AFFE_CHAR_MECA       :     0.04 :     0.02 :     0.06 :     0.05 *
* AFFE_CHAR_MECA       :     0.04 :     0.01 :     0.05 :     0.07 *
* AFFE_CHAR_MECA       :     0.04 :     0.02 :     0.06 :     0.05 *
* MECA_STATIQUE        :     3.28 :     0.39 :     3.67 :     3.79 *
* CALC_CHAMP           :     1.19 :     0.14 :     1.33 :     1.39 *
* IMPR_RESU            :     0.07 :     0.04 :     0.11 :     0.11 *
* IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.01 *
* MECA_STATIQUE        :     3.37 :     0.38 :     3.75 :     3.83 *
* CALC_CHAMP           :     1.13 :     0.14 :     1.27 :     1.29 *
* IMPR_RESU            :     0.01 :     0.01 :     0.02 :     0.02 *
* IMPR_RESU            :     0.02 :     0.00 :     0.02 :     0.01 *
* MECA_STATIQUE        :     3.33 :     0.37 :     3.70 :     3.76 *
* CALC_CHAMP           :     1.21 :     0.15 :     1.36 :     1.40 *
* IMPR_RESU            :     0.01 :     0.01 :     0.02 :     0.02 *
* IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.01 *
* MECA_STATIQUE        :     3.35 :     0.39 :     3.74 :     3.79 *
* CALC_CHAMP           :     1.16 :     0.14 :     1.30 :     1.34 *
* IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.01 *
* IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.02 *
* MECA_STATIQUE        :     3.32 :     0.38 :     3.70 :     3.76 *
* CALC_CHAMP           :     1.11 :     0.13 :     1.24 :     1.26 *
* IMPR_RESU            :     0.02 :     0.01 :     0.03 :     0.02 *
* IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.01 *
* MECA_STATIQUE        :     3.24 :     0.39 :     3.63 :     3.70 *
* CALC_CHAMP           :     1.19 :     0.14 :     1.33 :     1.35 *
* IMPR_RESU            :     0.02 :     0.00 :     0.02 :     0.02 *
* IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.02 *
* MECA_STATIQUE        :     3.25 :     0.37 :     3.62 :     3.68 *
* CALC_CHAMP           :     1.11 :     0.16 :     1.27 :     1.29 *
* IMPR_RESU            :     0.01 :     0.01 :     0.02 :     0.02 *
* IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.01 *
* FIN                  :     0.09 :     0.23 :     0.32 :     0.85 *
*  . part Superviseur  :     0.48 :     0.13 :     0.61 :     0.87 *
*  . part Fortran      :    32.54 :     4.19 :    36.73 :    38.08 *
********************************************************************************
* TOTAL_JOB            :    33.03 :     4.32 :    37.35 :    38.95 *
********************************************************************************
```

## Result file- Winged_Flange_Setup1of3_ Candidate 3

```
          -- CODE_ASTER -- VERSION : EXPLOITATION (stable) --

               Version 11.5.0 modifiÃ©e le 10/12/2013
                rÃ©vision 2b68bca7 - branche 'v11'
                  Copyright EDF R&D 1991 - 2014

              ExÃ©cution du : Wed Jun 18 15:23:43 2014
              Nom de la machine : thomas-VBox-Ubuntu14
                    Architecture : 64bit
                 Type de processeur : x86_64
            SystÃ¨me d'exploitation : Linux 3.13.0-24-generic
                 Langue des messages : de (UTF-8)

                   Version de Python : 2.7.3
                    Version de NumPy : 1.7.1
                   ParallÃ©lisme MPI : inactif
                  ParallÃ©lisme OpenMP : actif
                 Nombre de processus utilisÃ©s : 1
                Version de la librairie HDF5 : 1.8.10
                 Version de la librairie MED : 3.0.7
                Version de la librairie MUMPS : 4.10.0
               Version de la librairie SCOTCH : 5.1.10
              MÃ©moire limite pour l'exÃ©cution : 762.00 Mo
               consommÃ©e par l'initialisation : 213.67 Mo
              par les objets du jeu de commandes : 2.82 Mo
             reste pour l'allocation dynamique : 544.87 Mo
            Taille limite des fichiers d'Ã©change : 48.00 Go
  !-------------------------------------------------------------------------------!
  ! <A> <SUPERVIS_42>                                                             !
  !                                                                               !
  ! Les fichiers suivants ont Ã©tÃ© modifiÃ©s par rapport Ã  la derniÃ¨re rÃ©vision 2b68bca7 : !
  !                                                                               !
  ! checkout.log, configure.log, hg_diff.log, install.log, make.log               !
  !                                                                               !
  ! Ceci est une alarme. Si vous ne comprenez pas le sens de cette                !
  ! alarme, vous pouvez obtenir des rÃ©sultats inattendus !                        !
  !-------------------------------------------------------------------------------!
ASTER 11.05.00 CONCEPT SOLU_0 CALCULE LE 18/06/2014 A 15:23:45 DE TYPE
EVOL_ELAS

                ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface
------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -6.73943370271628E-06 EN    1 NOEUD(S) : N469
 LA VALEUR MAXIMALE DE DY      EST  9.10009785183327E-06 EN    1 NOEUD(S) : N384
 LA VALEUR MAXIMALE DE DZ      EST  2.10538121149455E-06 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DX      EST -2.46428189272013E-05 EN    1 NOEUD(S) : N38
 LA VALEUR MINIMALE DE DY      EST -5.81343066207557E-07 EN    1 NOEUD(S) : N36
```

```
 LA VALEUR MINIMALE DE DZ      EST -1.17299622671210E-04 EN    1 NOEUD(S) : N380

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_1 CALCULE LE 18/06/2014 A 15:23:50 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -8.78438741395694E-06 EN    1 NOEUD(S) : N37
 LA VALEUR MAXIMALE DE DY      EST  5.78448519128115E-06 EN    1 NOEUD(S) : N387
 LA VALEUR MAXIMALE DE DZ      EST  2.61688024058170E-06 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DX      EST -2.86587411576012E-05 EN    1 NOEUD(S) : N38
 LA VALEUR MINIMALE DE DY      EST -3.34330283487248E-06 EN    1 NOEUD(S) : N38
 LA VALEUR MINIMALE DE DZ      EST -1.20498644792121E-04 EN    1 NOEUD(S) : N380

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_2 CALCULE LE 18/06/2014 A 15:23:55 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.91574115201116E-05 EN    1 NOEUD(S) : N37
 LA VALEUR MAXIMALE DE DY      EST -1.50012923644509E-05 EN    1 NOEUD(S) : N37
 LA VALEUR MAXIMALE DE DZ      EST  8.76569357636116E-06 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DX      EST -9.79293422617070E-05 EN    1 NOEUD(S) : N290
 LA VALEUR MINIMALE DE DY      EST -8.45824886216387E-05 EN    1 NOEUD(S) : N284
 LA VALEUR MINIMALE DE DZ      EST -3.18894670228471E-04 EN    1 NOEUD(S) : N443

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_3 CALCULE LE 18/06/2014 A 15:24:00 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -6.47433297297150E-06 EN    1 NOEUD(S) : N469
 LA VALEUR MAXIMALE DE DY      EST  1.07023887329039E-05 EN    1 NOEUD(S) : N308
 LA VALEUR MAXIMALE DE DZ      EST  6.60245407114925E-06 EN    1 NOEUD(S) : N38
 LA VALEUR MINIMALE DE DX      EST -2.53869717113546E-05 EN    1 NOEUD(S) : N508
 LA VALEUR MINIMALE DE DY      EST -1.91991452130889E-07 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DZ      EST -1.10462124380365E-04 EN    1 NOEUD(S) : N381
```

```
 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_4 CALCULE LE 18/06/2014 A 15:24:04 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -6.61506280353246E-06 EN    1 NOEUD(S) : N469
 LA VALEUR MAXIMALE DE DY       EST  1.09813616770734E-05 EN    1 NOEUD(S) : N308
 LA VALEUR MAXIMALE DE DZ       EST  3.70688668387623E-06 EN    1 NOEUD(S) : N38
 LA VALEUR MINIMALE DE DX       EST -2.21060119354280E-05 EN    1 NOEUD(S) : N505
 LA VALEUR MINIMALE DE DY       EST -2.47035227157093E-07 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DZ       EST -1.08394172066074E-04 EN    1 NOEUD(S) : N380

 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_5 CALCULE LE 18/06/2014 A 15:24:09 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -7.17937736790883E-06 EN    1 NOEUD(S) : N469
 LA VALEUR MAXIMALE DE DY       EST  8.50610016685184E-06 EN    1 NOEUD(S) : N386
 LA VALEUR MAXIMALE DE DZ       EST  1.78460154668588E-06 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DX       EST -2.47210076983748E-05 EN    1 NOEUD(S) : N520
 LA VALEUR MINIMALE DE DY       EST -6.45800256962866E-07 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DZ       EST -1.04451731720700E-04 EN    1 NOEUD(S) : N382

 --------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_6 CALCULE LE 18/06/2014 A 15:24:14 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -7.80073173427364E-06 EN    1 NOEUD(S) : N469
 LA VALEUR MAXIMALE DE DY       EST  8.17902005398272E-06 EN    1 NOEUD(S) : N2445
 LA VALEUR MAXIMALE DE DZ       EST  1.73775735524381E-06 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DX       EST -2.25493992833467E-05 EN    1 NOEUD(S) : N38
 LA VALEUR MINIMALE DE DY       EST -5.56376119193795E-07 EN    1 NOEUD(S) : N36
 LA VALEUR MINIMALE DE DZ       EST -9.92473742946593E-05 EN    1 NOEUD(S) : N382
 <I> <FIN> FERMETURE DE LA BASE "GLOBALE" EFFECTUEE.

 <FIN> ArrÃªt normal dans "FIN".
```

```
 <I> <FIN> ARRET NORMAL DANS "FIN" PAR APPEL A "JEFINI".

 <I> <FIN> MEMOIRE JEVEUX MINIMALE REQUISE POUR L'EXECUTION :
58.96 Mo
 <I> <FIN> MEMOIRE JEVEUX OPTIMALE REQUISE POUR L'EXECUTION :
325.77 Mo
 <I> <FIN> MAXIMUM DE MEMOIRE UTILISEE PAR LE PROCESSUS LORS DE
L'EXECUTION :     730.89 Mo

 ********************************************************************************
 * COMMAND              :      USER :     SYSTEM :   USER+SYS :    ELAPSED *
 ********************************************************************************
 * init (jdc)           :      0.35 :      0.07 :      0.42 :      0.53 *
 *  . compile           :      0.01 :      0.00 :      0.01 :      0.01 *
 *  . exec_compile      :      0.12 :      0.04 :      0.16 :      0.23 *
 *  . report            :      0.04 :      0.00 :      0.04 :      0.03 *
 *  . build             :      0.00 :      0.00 :      0.00 :      0.00 *
 * DEBUT                :      0.05 :      0.03 :      0.08 :      0.16 *
 * DEFI_MATERIAU        :      0.00 :      0.01 :      0.01 :      0.01 *
 * DEFI_MATERIAU        :      0.01 :      0.00 :      0.01 :      0.01 *
 * LIRE_MAILLAGE        :      0.32 :      0.05 :      0.37 :      0.49 *
 * MODI_MAILLAGE        :      0.23 :      0.02 :      0.25 :      0.28 *
 * AFFE_MODELE          :      0.12 :      0.02 :      0.14 :      0.16 *
 * AFFE_MATERIAU        :      0.01 :      0.00 :      0.01 :      0.01 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.06 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.05 *
 * AFFE_CHAR_MECA       :      0.04 :      0.01 :      0.05 :      0.07 *
 * AFFE_CHAR_MECA       :      0.04 :      0.02 :      0.06 :      0.08 *
 * AFFE_CHAR_MECA       :      0.04 :      0.02 :      0.06 :      0.06 *
 * AFFE_CHAR_MECA       :      0.04 :      0.01 :      0.05 :      0.05 *
 * AFFE_CHAR_MECA       :      0.03 :      0.02 :      0.05 :      0.05 *
 * AFFE_CHAR_MECA       :      0.04 :      0.01 :      0.05 :      0.06 *
 * MECA_STATIQUE        :      3.08 :      0.38 :      3.46 :      3.60 *
 * CALC_CHAMP           :      1.19 :      0.15 :      1.34 :      1.36 *
 * IMPR_RESU            :      0.07 :      0.03 :      0.10 :      0.11 *
 * IMPR_RESU            :      0.01 :      0.01 :      0.02 :      0.03 *
 * MECA_STATIQUE        :      3.07 :      0.39 :      3.46 :      3.51 *
 * CALC_CHAMP           :      1.08 :      0.15 :      1.23 :      1.23 *
 * IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.02 *
 * IMPR_RESU            :      0.01 :      0.01 :      0.02 :      0.02 *
 * MECA_STATIQUE        :      3.12 :      0.37 :      3.49 :      3.54 *
 * CALC_CHAMP           :      1.16 :      0.14 :      1.30 :      1.32 *
 * IMPR_RESU            :      0.02 :      0.00 :      0.02 :      0.01 *
 * IMPR_RESU            :      0.01 :      0.01 :      0.02 :      0.01 *
 * MECA_STATIQUE        :      3.06 :      0.39 :      3.45 :      3.52 *
 * CALC_CHAMP           :      1.15 :      0.15 :      1.30 :      1.33 *
 * IMPR_RESU            :      0.02 :      0.00 :      0.02 :      0.02 *
 * IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.01 *
 * MECA_STATIQUE        :      3.11 :      0.36 :      3.47 :      3.53 *
 * CALC_CHAMP           :      1.12 :      0.15 :      1.27 :      1.29 *
 * IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.02 *
 * IMPR_RESU            :      0.01 :      0.01 :      0.02 :      0.02 *
 * MECA_STATIQUE        :      3.08 :      0.38 :      3.46 :      3.51 *
```

```
* CALC_CHAMP            :      1.16 :      0.14 :      1.30 :      1.33 *
* IMPR_RESU             :      0.01 :      0.01 :      0.02 :      0.02 *
* IMPR_RESU             :      0.01 :      0.00 :      0.01 :      0.02 *
* MECA_STATIQUE         :      3.13 :      0.48 :      3.61 :      3.68 *
* CALC_CHAMP            :      1.11 :      0.14 :      1.25 :      1.28 *
* IMPR_RESU             :      0.01 :      0.01 :      0.02 :      0.02 *
* IMPR_RESU             :      0.01 :      0.00 :      0.01 :      0.02 *
* FIN                   :      0.09 :      0.26 :      0.35 :      0.62 *
*  . part Superviseur    :      0.50 :      0.14 :      0.64 :      0.99 *
*  . part Fortran        :     30.84 :      4.31 :     35.15 :     36.26 *
********************************************************************************
* TOTAL_JOB             :     31.34 :      4.45 :     35.79 :     37.26 *
********************************************************************************
```

## Result file - Winged_Flange_Setup1of3_ Candidate 6

```
          -- CODE_ASTER -- VERSION : EXPLOITATION (stable) --

               Version 11.5.0 modifiÃ©e le 10/12/2013
                 rÃ©vision 2b68bca7 - branche 'v11'
                   Copyright EDF R&D 1991 - 2014

              ExÃ©cution du : Wed Jun 18 15:24:40 2014
              Nom de la machine : thomas-VBox-Ubuntu14
                    Architecture : 64bit
                 Type de processeur : x86_64
            SystÃ¨me d'exploitation : Linux 3.13.0-24-generic
                 Langue des messages : de (UTF-8)

                    Version de Python : 2.7.3
                    Version de NumPy : 1.7.1
                   ParallÃ©lisme MPI : inactif
                  ParallÃ©lisme OpenMP : actif
                 Nombre de processus utilisÃ©s : 1
                Version de la librairie HDF5 : 1.8.10
                 Version de la librairie MED : 3.0.7
                Version de la librairie MUMPS : 4.10.0
               Version de la librairie SCOTCH : 5.1.10
              MÃ©moire limite pour l'exÃ©cution : 762.00 Mo
               consommÃ©e par l'initialisation : 213.68 Mo
              par les objets du jeu de commandes : 2.86 Mo
             reste pour l'allocation dynamique : 544.82 Mo
            Taille limite des fichiers d'Ã©change : 48.00 Go

  !---------------------------------------------------------------------------!
  ! <A> <SUPERVIS_42>                                                         !
  !                                                                           !
  ! Les fichiers suivants ont Ã©tÃ© modifiÃ©s par rapport Ã  la derniÃ¨re rÃ©vision 2b68bca7 : !
  !                                                                           !
  ! checkout.log, configure.log, hg_diff.log, install.log, make.log           !
  !                                                                           !
  !                                                                           !
  ! Ceci est une alarme. Si vous ne comprenez pas le sens de cette            !
  ! alarme, vous pouvez obtenir des rÃ©sultats inattendus !                    !
  !---------------------------------------------------------------------------!
ASTER 11.05.00 CONCEPT SOLU_0 CALCULE LE 18/06/2014 A 15:24:42 DE TYPE
EVOL_ELAS

               ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.62618637862813E-06 EN    1 NOEUD(S) : N529
 LA VALEUR MAXIMALE DE DY      EST  3.67701815662881E-06 EN    1 NOEUD(S) : N340
```

```
 LA VALEUR MAXIMALE DE DZ      EST  1.34830903942242E-06 EN    1 NOEUD(S) : N41
 LA VALEUR MINIMALE DE DX      EST -8.91372763094604E-06 EN    1 NOEUD(S) : N316
 LA VALEUR MINIMALE DE DY      EST -5.60714820926965E-07 EN    1 NOEUD(S) : N386
 LA VALEUR MINIMALE DE DZ      EST -4.43698019685830E-05 EN    1 NOEUD(S) : N413

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_1 CALCULE LE 18/06/2014 A 15:24:47 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -2.60275926795810E-06 EN    1 NOEUD(S) : N39
 LA VALEUR MAXIMALE DE DY      EST  1.50023666687153E-06 EN    1 NOEUD(S) : N339
 LA VALEUR MAXIMALE DE DZ      EST  3.08072252925533E-07 EN    1 NOEUD(S) : N426
 LA VALEUR MINIMALE DE DX      EST -1.00368141118938E-05 EN    1 NOEUD(S) : N315
 LA VALEUR MINIMALE DE DY      EST -1.85663710309989E-06 EN    1 NOEUD(S) : N400
 LA VALEUR MINIMALE DE DZ      EST -4.40397638775711E-05 EN    1 NOEUD(S) : N414

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_2 CALCULE LE 18/06/2014 A 15:24:53 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST  9.01510979135357E-08 EN    1 NOEUD(S) : N40
 LA VALEUR MAXIMALE DE DY      EST -6.01529249234788E-06 EN    1 NOEUD(S) : N438
 LA VALEUR MAXIMALE DE DZ      EST -8.45528352255530E-08 EN    1 NOEUD(S) : N39
 LA VALEUR MINIMALE DE DX      EST -5.90336203197366E-05 EN    1 NOEUD(S) : N323
 LA VALEUR MINIMALE DE DY      EST -6.89220548097154E-05 EN    1 NOEUD(S) : N319
 LA VALEUR MINIMALE DE DZ      EST -1.32144825945765E-04 EN    1 NOEUD(S) : N476

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_3 CALCULE LE 18/06/2014 A 15:24:58 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.90256640174932E-06 EN    1 NOEUD(S) : N39
 LA VALEUR MAXIMALE DE DY      EST  4.41011112492410E-06 EN    1 NOEUD(S) : N339
 LA VALEUR MAXIMALE DE DZ      EST  2.03531938003154E-06 EN    1 NOEUD(S) : N41
 LA VALEUR MINIMALE DE DX      EST -1.19402865476445E-05 EN    1 NOEUD(S) : N541
```

```
 LA VALEUR MINIMALE DE DY      EST -4.36768608008364E-07 EN    1 NOEUD(S) : N385
 LA VALEUR MINIMALE DE DZ      EST -4.07936203130662E-05 EN    1 NOEUD(S) : N413

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_4 CALCULE LE 18/06/2014 A 15:25:03 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.85966580075577E-06 EN    1 NOEUD(S) : N39
 LA VALEUR MAXIMALE DE DY      EST  4.87952900047684E-06 EN    1 NOEUD(S) : N4764
 LA VALEUR MAXIMALE DE DZ      EST  1.44540959893391E-06 EN    1 NOEUD(S) : N41
 LA VALEUR MINIMALE DE DX      EST -1.18280034530436E-05 EN    1 NOEUD(S) : N538
 LA VALEUR MINIMALE DE DY      EST -4.33095731456706E-07 EN    1 NOEUD(S) : N385
 LA VALEUR MINIMALE DE DZ      EST -3.89443544928295E-05 EN    1 NOEUD(S) : N413

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_5 CALCULE LE 18/06/2014 A 15:25:09 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.54540856219016E-06 EN    1 NOEUD(S) : N39
 LA VALEUR MAXIMALE DE DY      EST  3.36448209183762E-06 EN    1 NOEUD(S) : N340
 LA VALEUR MAXIMALE DE DZ      EST  9.36013525484139E-07 EN    1 NOEUD(S) : N41
 LA VALEUR MINIMALE DE DX      EST -1.29277857134410E-05 EN    1 NOEUD(S) : N553
 LA VALEUR MINIMALE DE DY      EST -9.11360921977849E-07 EN    1 NOEUD(S) : N401
 LA VALEUR MINIMALE DE DZ      EST -3.71145180437889E-05 EN    1 NOEUD(S) : N415

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_6 CALCULE LE 18/06/2014 A 15:25:14 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.65187618733654E-06 EN    1 NOEUD(S) : N39
 LA VALEUR MAXIMALE DE DY      EST  3.21969137289899E-06 EN    1 NOEUD(S) : N339
 LA VALEUR MAXIMALE DE DZ      EST  8.67392292704589E-07 EN    1 NOEUD(S) : N41
 LA VALEUR MINIMALE DE DX      EST -1.31970713576565E-05 EN    1 NOEUD(S) : N547
 LA VALEUR MINIMALE DE DY      EST -4.84799103657023E-07 EN    1 NOEUD(S) : N387
 LA VALEUR MINIMALE DE DZ      EST -3.39526628643046E-05 EN    1 NOEUD(S) : N471
```

```
 <I> <FIN> FERMETURE DE LA BASE "GLOBALE" EFFECTUEE.

 <FIN> ArrÃªt normal dans "FIN".
 <I> <FIN> ARRET NORMAL DANS "FIN" PAR APPEL A "JEFINI".

 <I> <FIN> MEMOIRE JEVEUX MINIMALE REQUISE POUR L'EXECUTION :
60.41 Mo
 <I> <FIN> MEMOIRE JEVEUX OPTIMALE REQUISE POUR L'EXECUTION :
333.78 Mo
 <I> <FIN> MAXIMUM DE MEMOIRE UTILISEE PAR LE PROCESSUS LORS DE
L'EXECUTION :     732.72 Mo

 ********************************************************************************
 * COMMAND              :     USER :    SYSTEM :  USER+SYS :   ELAPSED *
 ********************************************************************************
 * init (jdc)           :     0.36 :     0.08 :     0.44 :     0.54 *
 *  . compile           :     0.00 :     0.00 :     0.00 :     0.00 *
 *  . exec_compile      :     0.13 :     0.04 :     0.17 :     0.22 *
 *  . report            :     0.04 :     0.00 :     0.04 :     0.04 *
 *  . build             :     0.00 :     0.00 :     0.00 :     0.00 *
 * DEBUT                :     0.05 :     0.04 :     0.09 :     0.17 *
 * DEFI_MATERIAU        :     0.00 :     0.00 :     0.00 :     0.02 *
 * DEFI_MATERIAU        :     0.01 :     0.00 :     0.01 :     0.02 *
 * LIRE_MAILLAGE        :     0.34 :     0.06 :     0.40 :     0.63 *
 * MODI_MAILLAGE        :     0.25 :     0.02 :     0.27 :     0.31 *
 * AFFE_MODELE          :     0.14 :     0.02 :     0.16 :     0.18 *
 * AFFE_MATERIAU        :     0.01 :     0.01 :     0.02 :     0.02 *
 * AFFE_CHAR_MECA       :     0.04 :     0.01 :     0.05 :     0.05 *
 * AFFE_CHAR_MECA       :     0.03 :     0.01 :     0.04 :     0.04 *
 * AFFE_CHAR_MECA       :     0.04 :     0.01 :     0.05 :     0.05 *
 * AFFE_CHAR_MECA       :     0.05 :     0.02 :     0.07 :     0.08 *
 * AFFE_CHAR_MECA       :     0.03 :     0.02 :     0.05 :     0.05 *
 * AFFE_CHAR_MECA       :     0.04 :     0.01 :     0.05 :     0.06 *
 * AFFE_CHAR_MECA       :     0.04 :     0.02 :     0.06 :     0.07 *
 * AFFE_CHAR_MECA       :     0.04 :     0.02 :     0.06 :     0.08 *
 * MECA_STATIQUE        :     3.41 :     0.41 :     3.82 :     4.28 *
 * CALC_CHAMP           :     1.19 :     0.15 :     1.34 :     1.39 *
 * IMPR_RESU            :     0.06 :     0.03 :     0.09 :     0.11 *
 * IMPR_RESU            :     0.02 :     0.01 :     0.03 :     0.01 *
 * MECA_STATIQUE        :     3.37 :     0.40 :     3.77 :     4.06 *
 * CALC_CHAMP           :     1.24 :     0.16 :     1.40 :     1.50 *
 * IMPR_RESU            :     0.01 :     0.01 :     0.02 :     0.02 *
 * IMPR_RESU            :     0.02 :     0.00 :     0.02 :     0.02 *
 * MECA_STATIQUE        :     3.35 :     0.39 :     3.74 :     3.84 *
 * CALC_CHAMP           :     1.19 :     0.15 :     1.34 :     1.35 *
 * IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.02 *
 * IMPR_RESU            :     0.01 :     0.01 :     0.02 :     0.02 *
 * MECA_STATIQUE        :     3.24 :     0.38 :     3.62 :     3.75 *
 * CALC_CHAMP           :     1.14 :     0.16 :     1.30 :     1.31 *
 * IMPR_RESU            :     0.01 :     0.00 :     0.01 :     0.02 *
 * IMPR_RESU            :     0.01 :     0.01 :     0.02 :     0.02 *
 * MECA_STATIQUE        :     3.38 :     0.40 :     3.78 :     3.88 *
 * CALC_CHAMP           :     1.22 :     0.15 :     1.37 :     1.42 *
```

```
* IMPR_RESU            :      0.01 :      0.01 :      0.02 :      0.02 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.02 *
* MECA_STATIQUE        :      3.30 :      0.40 :      3.70 :      3.95 *
* CALC_CHAMP           :      1.23 :      0.19 :      1.42 :      1.47 *
* IMPR_RESU            :      0.02 :      0.00 :      0.02 :      0.03 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.02 *
* MECA_STATIQUE        :      3.27 :      0.42 :      3.69 :      3.82 *
* CALC_CHAMP           :      1.11 :      0.15 :      1.26 :      1.29 *
* IMPR_RESU            :      0.01 :      0.01 :      0.02 :      0.01 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.02 *
* FIN                  :      0.08 :      0.27 :      0.35 :      0.50 *
*  . part Superviseur  :      0.49 :      0.14 :      0.63 :      1.00 *
*  . part Fortran      :     32.93 :      4.50 :     37.43 :     39.66 *
********************************************************************************
* TOTAL_JOB            :     33.44 :      4.64 :     38.08 :     40.66 *
********************************************************************************
```

## Result file - Cyl_D60_H15_Setup1of1_ Candidate 1

```
          -- CODE_ASTER -- VERSION : EXPLOITATION (stable) --

               Version 11.5.0 modifiÃ©e le 10/12/2013
                rÃ©vision 2b68bca7 - branche 'v11'
                  Copyright EDF R&D 1991 - 2014

              ExÃ©cution du : Sun Jun 15 12:55:53 2014
              Nom de la machine : thomas-VBox-Ubuntu14
                     Architecture : 64bit
                 Type de processeur : x86_64
            SystÃ¨me d'exploitation : Linux 3.13.0-24-generic
                 Langue des messages : de (UTF-8)

                    Version de Python : 2.7.3
                     Version de NumPy : 1.7.1
                    ParallÃ©lisme MPI : inactif
                   ParallÃ©lisme OpenMP : actif
                  Nombre de processus utilisÃ©s : 1
                 Version de la librairie HDF5 : 1.8.10
                  Version de la librairie MED : 3.0.7
                 Version de la librairie MUMPS : 4.10.0
                Version de la librairie SCOTCH : 5.1.10
               MÃ©moire limite pour l'exÃ©cution : 762.00 Mo
                consommÃ©e par l'initialisation : 213.61 Mo
               par les objets du jeu de commandes : 2.66 Mo
              reste pour l'allocation dynamique : 545.34 Mo
             Taille limite des fichiers d'Ã©change : 48.00 Go

  !---------------------------------------------------------------------------------!
  ! <A> <SUPERVIS_42>                                                               !
  !                                                                                 !
  ! Les fichiers suivants ont Ã©tÃ© modifiÃ©s par rapport Ã  la derniÃ¨re rÃ©vision 2b68bca7 : !
  !                                                                                 !
  ! checkout.log, configure.log, hg_diff.log, install.log, make.log                 !
  !                                                                                 !
  ! Ceci est une alarme. Si vous ne comprenez pas le sens de cette                  !
  ! alarme, vous pouvez obtenir des rÃ©sultats inattendus !                          !
  !---------------------------------------------------------------------------------!
ASTER 11.05.00 CONCEPT SOLU_0 CALCULE LE 15/06/2014 A 12:55:53 DE TYPE
EVOL_ELAS

               ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.02651828072531E-05 EN    1 NOEUD(S) : N425
 LA VALEUR MAXIMALE DE DY      EST  1.90216875111116E-06 EN    1 NOEUD(S) : N554
```

```
 LA VALEUR MAXIMALE DE DZ      EST  1.57765977923553E-06 EN    1 NOEUD(S) : N558
 LA VALEUR MINIMALE DE DX      EST -3.89140490825472E-05 EN    1 NOEUD(S) : N525
 LA VALEUR MINIMALE DE DY      EST -2.31755450035287E-05 EN    1 NOEUD(S) : N452
 LA VALEUR MINIMALE DE DZ      EST -7.41220905572222E-05 EN    1 NOEUD(S) : N438

 ---------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_1 CALCULE LE 15/06/2014 A 12:55:55 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.04975089425744E-04 EN    1 NOEUD(S) : N505
 LA VALEUR MAXIMALE DE DY      EST -8.30282455034683E-05 EN    1 NOEUD(S) : N424
 LA VALEUR MAXIMALE DE DZ      EST  4.26577428881996E-05 EN    1 NOEUD(S) : N558
 LA VALEUR MINIMALE DE DX      EST -3.28615954697852E-04 EN    1 NOEUD(S) : N651
 LA VALEUR MINIMALE DE DY      EST -2.74223073653641E-04 EN    1 NOEUD(S) : N576
 LA VALEUR MINIMALE DE DZ      EST -5.67065028995973E-04 EN    1 NOEUD(S) : N577

 ---------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_2 CALCULE LE 15/06/2014 A 12:55:57 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.15364563613713E-05 EN    1 NOEUD(S) : N1836
 LA VALEUR MAXIMALE DE DY      EST  4.04050322729092E-07 EN    1 NOEUD(S) : N554
 LA VALEUR MAXIMALE DE DZ      EST  1.72635897053468E-06 EN    1 NOEUD(S) : N558
 LA VALEUR MINIMALE DE DX      EST -5.05470260462416E-05 EN    1 NOEUD(S) : N525
 LA VALEUR MINIMALE DE DY      EST -2.80235037059224E-05 EN    1 NOEUD(S) : N498
 LA VALEUR MINIMALE DE DZ      EST -8.95658916699613E-05 EN    1 NOEUD(S) : N438

 ---------------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_3 CALCULE LE 15/06/2014 A 12:55:58 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.15774065809983E-05 EN    1 NOEUD(S) : N1836
 LA VALEUR MAXIMALE DE DY      EST  9.66534540997147E-07 EN    1 NOEUD(S) : N554
 LA VALEUR MAXIMALE DE DZ      EST  1.76528307661667E-06 EN    1 NOEUD(S) : N558
 LA VALEUR MINIMALE DE DX      EST -4.86386023193841E-05 EN    1 NOEUD(S) : N524
```

```
 LA VALEUR MINIMALE DE DY       EST -3.00113141419681E-05 EN    1 NOEUD(S) : N498
 LA VALEUR MINIMALE DE DZ       EST -8.79361244224870E-05 EN    1 NOEUD(S) : N438

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_4 CALCULE LE 15/06/2014 A 12:56:00 DE TYPE
EVOL_ELAS

                ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -7.50030364889179E-06 EN    1 NOEUD(S) : N1836
 LA VALEUR MAXIMALE DE DY       EST  1.03761663885977E-06 EN    1 NOEUD(S) : N554
 LA VALEUR MAXIMALE DE DZ       EST  1.22627559187890E-06 EN    1 NOEUD(S) : N558
 LA VALEUR MINIMALE DE DX       EST -4.34331995008092E-05 EN    1 NOEUD(S) : N467
 LA VALEUR MINIMALE DE DY       EST -2.38576294789665E-05 EN    1 NOEUD(S) : N453
 LA VALEUR MINIMALE DE DZ       EST -5.03560593424477E-05 EN    1 NOEUD(S) : N67
 <I> <FIN> FERMETURE DE LA BASE "GLOBALE" EFFECTUEE.

 <FIN> ArrÃªt normal dans "FIN".
 <I> <FIN> ARRET NORMAL DANS "FIN" PAR APPEL A "JEFINI".

 <I> <FIN> MEMOIRE JEVEUX MINIMALE REQUISE POUR L'EXECUTION :
30.70 Mo
 <I> <FIN> MEMOIRE JEVEUX OPTIMALE REQUISE POUR L'EXECUTION :
113.39 Mo
 <I> <FIN> MAXIMUM DE MEMOIRE UTILISEE PAR LE PROCESSUS LORS DE
L'EXECUTION :     710.27 Mo

 ********************************************************************************
 * COMMAND              :      USER :     SYSTEM :   USER+SYS :    ELAPSED *
 ********************************************************************************
 * init (jdc)           :      0.29 :      0.06 :      0.35 :      0.45 *
 *  . compile           :      0.00 :      0.00 :      0.00 :      0.00 *
 *  . exec_compile      :      0.10 :      0.03 :      0.13 :      0.18 *
 *  . report            :      0.02 :      0.00 :      0.02 :      0.02 *
 *  . build             :      0.00 :      0.00 :      0.00 :      0.00 *
 * DEBUT                :      0.04 :      0.03 :      0.07 :      0.10 *
 * DEFI_MATERIAU        :      0.00 :      0.00 :      0.00 :      0.01 *
 * DEFI_MATERIAU        :      0.01 :      0.00 :      0.01 :      0.01 *
 * LIRE_MAILLAGE        :      0.12 :      0.02 :      0.14 :      0.18 *
 * MODI_MAILLAGE        :      0.07 :      0.01 :      0.08 :      0.09 *
 * AFFE_MODELE          :      0.04 :      0.01 :      0.05 :      0.09 *
 * AFFE_MATERIAU        :      0.01 :      0.01 :      0.02 :      0.01 *
 * AFFE_CHAR_MECA       :      0.03 :      0.00 :      0.03 :      0.05 *
 * AFFE_CHAR_MECA       :      0.01 :      0.01 :      0.02 :      0.03 *
 * AFFE_CHAR_MECA       :      0.02 :      0.01 :      0.03 :      0.04 *
 * AFFE_CHAR_MECA       :      0.02 :      0.01 :      0.03 :      0.03 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.04 *
 * AFFE_CHAR_MECA       :      0.02 :      0.01 :      0.03 :      0.03 *
 * MECA_STATIQUE        :      0.89 :      0.14 :      1.03 :      1.13 *
```

```
* CALC_CHAMP          :      0.41 :      0.06 :      0.47 :      0.48 *
* IMPR_RESU           :      0.05 :      0.03 :      0.08 :      0.10 *
* IMPR_RESU           :      0.01 :      0.00 :      0.01 :      0.00 *
* MECA_STATIQUE       :      0.86 :      0.13 :      0.99 :      1.04 *
* CALC_CHAMP          :      0.43 :      0.06 :      0.49 :      0.50 *
* IMPR_RESU           :      0.01 :      0.01 :      0.02 :      0.01 *
* IMPR_RESU           :      0.01 :      0.00 :      0.01 :      0.01 *
* MECA_STATIQUE       :      0.87 :      0.13 :      1.00 :      1.00 *
* CALC_CHAMP          :      0.41 :      0.07 :      0.48 :      0.49 *
* IMPR_RESU           :      0.01 :      0.00 :      0.01 :      0.01 *
* IMPR_RESU           :      0.01 :      0.00 :      0.01 :      0.02 *
* MECA_STATIQUE       :      0.89 :      0.13 :      1.02 :      1.06 *
* CALC_CHAMP          :      0.40 :      0.06 :      0.46 :      0.46 *
* IMPR_RESU           :      0.01 :      0.01 :      0.02 :      0.02 *
* IMPR_RESU           :      0.01 :      0.00 :      0.01 :      0.01 *
* MECA_STATIQUE       :      0.90 :      0.13 :      1.03 :      1.05 *
* CALC_CHAMP          :      0.40 :      0.07 :      0.47 :      0.47 *
* IMPR_RESU           :      0.01 :      0.00 :      0.01 :      0.01 *
* IMPR_RESU           :      0.00 :      0.00 :      0.00 :      0.01 *
* FIN                 :      0.04 :      0.08 :      0.12 :      0.13 *
*  . part Superviseur :      0.38 :      0.12 :      0.50 :      0.74 *
*  . part Fortran     :      6.98 :      1.20 :      8.18 :      8.52 *
********************************************************************************
* TOTAL_JOB           :      7.36 :      1.32 :      8.68 :      9.27 *
********************************************************************************
```

## Result file - Cyl_D60_H15_Setup1of1_ Candidate 5

```
          -- CODE_ASTER -- VERSION : EXPLOITATION (stable) --

               Version 11.5.0 modifiÃ©e le 10/12/2013
                 rÃ©vision 2b68bca7 - branche 'v11'
                   Copyright EDF R&D 1991 - 2014

              ExÃ©cution du : Sat Jun 14 23:25:08 2014
              Nom de la machine : thomas-VBox-Ubuntu14
                    Architecture : 64bit
                 Type de processeur : x86_64
            SystÃ¨me d'exploitation : Linux 3.13.0-24-generic
                 Langue des messages : de (UTF-8)

                    Version de Python : 2.7.3
                     Version de NumPy : 1.7.1
                   ParallÃ©lisme MPI : inactif
                  ParallÃ©lisme OpenMP : actif
                Nombre de processus utilisÃ©s : 1
               Version de la librairie HDF5 : 1.8.10
                Version de la librairie MED : 3.0.7
               Version de la librairie MUMPS : 4.10.0
              Version de la librairie SCOTCH : 5.1.10
             MÃ©moire limite pour l'exÃ©cution : 762.00 Mo
              consommÃ©e par l'initialisation : 213.67 Mo
             par les objets du jeu de commandes : 2.60 Mo
            reste pour l'allocation dynamique : 545.35 Mo
           Taille limite des fichiers d'Ã©change : 48.00 Go
  !-------------------------------------------------------------------------------!
  ! <A> <SUPERVIS_42>                                                             !
  !                                                                               !
  ! Les fichiers suivants ont Ã©tÃ© modifiÃ©s par rapport Ã  la derniÃ¨re rÃ©vision 2b68bca7 : !
  !                                                                               !
  ! checkout.log, configure.log, hg_diff.log, install.log, make.log               !
  !                                                                               !
  !                                                                               !
  ! Ceci est une alarme. Si vous ne comprenez pas le sens de cette                !
  ! alarme, vous pouvez obtenir des rÃ©sultats inattendus !                        !
  !-------------------------------------------------------------------------------!
ASTER 11.05.00 CONCEPT SOLU_0 CALCULE LE 14/06/2014 A 23:25:09 DE TYPE
EVOL_ELAS

               ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -2.03801188587013E-05 EN    1 NOEUD(S) : N4293
 LA VALEUR MAXIMALE DE DY      EST  1.56393755136601E-06 EN    1 NOEUD(S) : N656
 LA VALEUR MAXIMALE DE DZ      EST  5.88964025130081E-06 EN    1 NOEUD(S) : N660
```

```
 LA VALEUR MINIMALE DE DX      EST -6.20428641216980E-05 EN    1 NOEUD(S) : N650
 LA VALEUR MINIMALE DE DY      EST -1.79896703496954E-05 EN    1 NOEUD(S) : N710
 LA VALEUR MINIMALE DE DZ      EST -1.17596427603760E-04 EN    1 NOEUD(S) : N549

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_1 CALCULE LE 14/06/2014 A 23:25:11 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.48397123336964E-04 EN    1 NOEUD(S) : N63
 LA VALEUR MAXIMALE DE DY      EST -1.28517133336483E-04 EN    1 NOEUD(S) : N533
 LA VALEUR MAXIMALE DE DZ      EST  9.78233420838651E-05 EN    1 NOEUD(S) : N658
 LA VALEUR MINIMALE DE DX      EST -5.93433727764895E-04 EN    1 NOEUD(S) : N750
 LA VALEUR MINIMALE DE DY      EST -3.50532989765877E-04 EN    1 NOEUD(S) : N678
 LA VALEUR MINIMALE DE DZ      EST -8.94017667115555E-04 EN    1 NOEUD(S) : N679

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_2 CALCULE LE 14/06/2014 A 23:25:13 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -2.51472293561466E-05 EN    1 NOEUD(S) : N63
 LA VALEUR MAXIMALE DE DY      EST -7.13062538220497E-07 EN    1 NOEUD(S) : N529
 LA VALEUR MAXIMALE DE DZ      EST  6.70743346272300E-06 EN    1 NOEUD(S) : N660
 LA VALEUR MINIMALE DE DX      EST -7.78279085369202E-05 EN    1 NOEUD(S) : N650
 LA VALEUR MINIMALE DE DY      EST -2.53093504506141E-05 EN    1 NOEUD(S) : N709
 LA VALEUR MINIMALE DE DZ      EST -1.37415756303950E-04 EN    1 NOEUD(S) : N548

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_3 CALCULE LE 14/06/2014 A 23:25:15 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -2.64005528406037E-05 EN    1 NOEUD(S) : N63
 LA VALEUR MAXIMALE DE DY      EST -3.07245052327645E-07 EN    1 NOEUD(S) : N656
 LA VALEUR MAXIMALE DE DZ      EST  6.62934291382695E-06 EN    1 NOEUD(S) : N660
 LA VALEUR MINIMALE DE DX      EST -7.46923847346391E-05 EN    1 NOEUD(S) : N64
 LA VALEUR MINIMALE DE DY      EST -2.76612642358497E-05 EN    1 NOEUD(S) : N709
```

```
 LA VALEUR MINIMALE DE DZ       EST -1.35009622564486E-04 EN    1 NOEUD(S) : N548

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_4 CALCULE LE 14/06/2014 A 23:25:18 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX       EST -2.31193924754524E-05 EN    1 NOEUD(S) : N4327
 LA VALEUR MAXIMALE DE DY       EST  1.27216351779463E-06 EN    1 NOEUD(S) : N529
 LA VALEUR MAXIMALE DE DZ       EST  4.66221820688529E-06 EN    1 NOEUD(S) : N660
 LA VALEUR MINIMALE DE DX       EST -5.89146819583194E-05 EN    1 NOEUD(S) : N577
 LA VALEUR MINIMALE DE DY       EST -1.76708800859821E-05 EN    1 NOEUD(S) : N710
 LA VALEUR MINIMALE DE DZ       EST -8.59049764418241E-05 EN    1 NOEUD(S) : N633
 <I> <FIN> FERMETURE DE LA BASE "GLOBALE" EFFECTUEE.

 <FIN> ArrÃªt normal dans "FIN".
 <I> <FIN> ARRET NORMAL DANS "FIN" PAR APPEL A "JEFINI".

 <I> <FIN> MEMOIRE JEVEUX MINIMALE REQUISE POUR L'EXECUTION :
35.15 Mo
 <I> <FIN> MEMOIRE JEVEUX OPTIMALE REQUISE POUR L'EXECUTION :
145.04 Mo
 <I> <FIN> MAXIMUM DE MEMOIRE UTILISEE PAR LE PROCESSUS LORS DE
L'EXECUTION :     712.73 Mo

 ********************************************************************************
 * COMMAND              :      USER :     SYSTEM :   USER+SYS :    ELAPSED *
 ********************************************************************************
 * init (jdc)           :      0.30 :      0.06 :      0.36 :      0.41 *
 *  . compile           :      0.00 :      0.00 :      0.00 :      0.00 *
 *  . exec_compile      :      0.11 :      0.03 :      0.14 :      0.17 *
 *  . report            :      0.02 :      0.00 :      0.02 :      0.02 *
 *  . build             :      0.00 :      0.00 :      0.00 :      0.00 *
 * DEBUT                :      0.05 :      0.03 :      0.08 :      0.14 *
 * DEFI_MATERIAU        :      0.00 :      0.01 :      0.01 :      0.01 *
 * DEFI_MATERIAU        :      0.01 :      0.00 :      0.01 :      0.01 *
 * LIRE_MAILLAGE        :      0.14 :      0.02 :      0.16 :      0.21 *
 * MODI_MAILLAGE        :      0.10 :      0.01 :      0.11 :      0.11 *
 * AFFE_MODELE          :      0.06 :      0.01 :      0.07 :      0.08 *
 * AFFE_MATERIAU        :      0.01 :      0.00 :      0.01 :      0.01 *
 * AFFE_CHAR_MECA       :      0.06 :      0.01 :      0.07 :      0.07 *
 * AFFE_CHAR_MECA       :      0.02 :      0.01 :      0.03 :      0.02 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.05 *
 * AFFE_CHAR_MECA       :      0.03 :      0.02 :      0.05 :      0.04 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.05 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.05 *
 * MECA_STATIQUE        :      1.28 :      0.24 :      1.52 :      1.62 *
 * CALC_CHAMP           :      0.55 :      0.10 :      0.65 :      0.66 *
```

```
* IMPR_RESU          :     0.04 :     0.02 :     0.06 :     0.06 *
* IMPR_RESU          :     0.01 :     0.01 :     0.02 :     0.01 *
* MECA_STATIQUE      :     1.30 :     0.23 :     1.53 :     1.57 *
* CALC_CHAMP         :     0.55 :     0.10 :     0.65 :     0.66 *
* IMPR_RESU          :     0.00 :     0.00 :     0.00 :     0.01 *
* IMPR_RESU          :     0.01 :     0.00 :     0.01 :     0.01 *
* MECA_STATIQUE      :     1.30 :     0.23 :     1.53 :     1.55 *
* CALC_CHAMP         :     0.58 :     0.10 :     0.68 :     0.70 *
* IMPR_RESU          :     0.00 :     0.00 :     0.00 :     0.01 *
* IMPR_RESU          :     0.01 :     0.01 :     0.02 :     0.01 *
* MECA_STATIQUE      :     1.29 :     0.25 :     1.54 :     1.56 *
* CALC_CHAMP         :     0.60 :     0.10 :     0.70 :     0.74 *
* IMPR_RESU          :     0.01 :     0.01 :     0.02 :     0.02 *
* IMPR_RESU          :     0.00 :     0.00 :     0.00 :     0.00 *
* MECA_STATIQUE      :     1.25 :     0.22 :     1.47 :     1.50 *
* CALC_CHAMP         :     0.59 :     0.11 :     0.70 :     0.73 *
* IMPR_RESU          :     0.01 :     0.00 :     0.01 :     0.01 *
* IMPR_RESU          :     0.01 :     0.00 :     0.01 :     0.01 *
* FIN                :     0.04 :     0.13 :     0.17 :     0.37 *
*  . part Superviseur :     0.43 :     0.10 :     0.53 :     0.67 *
*  . part Fortran     :     9.91 :     1.98 :    11.89 :    12.46 *
********************************************************************************
* TOTAL_JOB          :    10.34 :     2.08 :    12.42 :    13.13 *
********************************************************************************
```

## Result file - Cyl_D60_H15_Setup1of1_ Candidate 7

```
          -- CODE_ASTER -- VERSION : EXPLOITATION (stable) --

               Version 11.5.0 modifiÃ©e le 10/12/2013
                 rÃ©vision 2b68bca7 - branche 'v11'
                   Copyright EDF R&D 1991 - 2014

              ExÃ©cution du : Sun Jun 15 00:10:50 2014
              Nom de la machine : thomas-VBox-Ubuntu14
                     Architecture : 64bit
                 Type de processeur : x86_64
            SystÃ¨me d'exploitation : Linux 3.13.0-24-generic
                 Langue des messages : de (UTF-8)

                    Version de Python : 2.7.3
                     Version de NumPy : 1.7.1
                   ParallÃ©lisme MPI : inactif
                  ParallÃ©lisme OpenMP : actif
                Nombre de processus utilisÃ©s : 1
               Version de la librairie HDF5 : 1.8.10
                Version de la librairie MED : 3.0.7
               Version de la librairie MUMPS : 4.10.0
              Version de la librairie SCOTCH : 5.1.10
             MÃ©moire limite pour l'exÃ©cution : 762.00 Mo
              consommÃ©e par l'initialisation : 213.67 Mo
             par les objets du jeu de commandes : 2.55 Mo
            reste pour l'allocation dynamique : 545.39 Mo
           Taille limite des fichiers d'Ã©change : 48.00 Go
  !---------------------------------------------------------------------------------!
  ! <A> <SUPERVIS_42>                                                               !
  !                                                                                 !
  ! Les fichiers suivants ont Ã©tÃ© modifiÃ©s par rapport Ã  la derniÃ¨re rÃ©vision 2b68bca7 : !
  !                                                                                 !
  ! checkout.log, configure.log, hg_diff.log, install.log, make.log                 !
  !                                                                                 !
  !                                                                                 !
  ! Ceci est une alarme. Si vous ne comprenez pas le sens de cette                    !
  ! alarme, vous pouvez obtenir des rÃ©sultats inattendus !                          !
  !---------------------------------------------------------------------------------!
ASTER 11.05.00 CONCEPT SOLU_0 CALCULE LE 15/06/2014 A 00:10:51 DE TYPE
EVOL_ELAS

               ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.45624311004772E-05 EN    1 NOEUD(S) : N622
 LA VALEUR MAXIMALE DE DY      EST  1.26978812595281E-06 EN    1 NOEUD(S) : N559
 LA VALEUR MAXIMALE DE DZ      EST  3.72221680236681E-06 EN    1 NOEUD(S) : N564
```

```
 LA VALEUR MINIMALE DE DX      EST -5.04979705163130E-05 EN    1 NOEUD(S) : N528
 LA VALEUR MINIMALE DE DY      EST -4.21742795664892E-05 EN    1 NOEUD(S) : N457
 LA VALEUR MINIMALE DE DZ      EST -1.00058943578454E-04 EN    1 NOEUD(S) : N440

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_1 CALCULE LE 15/06/2014 A 00:10:53 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.03062690640653E-04 EN    1 NOEUD(S) : N3718
 LA VALEUR MAXIMALE DE DY      EST -1.00156653940629E-04 EN    1 NOEUD(S) : N426
 LA VALEUR MAXIMALE DE DZ      EST  7.28833503869066E-05 EN    1 NOEUD(S) : N563
 LA VALEUR MINIMALE DE DX      EST -3.87178017627374E-04 EN    1 NOEUD(S) : N652
 LA VALEUR MINIMALE DE DY      EST -3.18912411546841E-04 EN    1 NOEUD(S) : N582
 LA VALEUR MINIMALE DE DZ      EST -6.86059873870567E-04 EN    1 NOEUD(S) : N583

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_2 CALCULE LE 15/06/2014 A 00:10:54 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.86701176646232E-05 EN    1 NOEUD(S) : N622
 LA VALEUR MAXIMALE DE DY      EST -8.58081692745753E-07 EN    1 NOEUD(S) : N559
 LA VALEUR MAXIMALE DE DZ      EST  4.09530596373422E-06 EN    1 NOEUD(S) : N564
 LA VALEUR MINIMALE DE DX      EST -6.15421701906999E-05 EN    1 NOEUD(S) : N530
 LA VALEUR MINIMALE DE DY      EST -4.89491980534903E-05 EN    1 NOEUD(S) : N457
 LA VALEUR MINIMALE DE DZ      EST -1.15795361446576E-04 EN    1 NOEUD(S) : N440

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_3 CALCULE LE 15/06/2014 A 00:10:56 DE TYPE
EVOL_ELAS

            ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -2.03610340827034E-05 EN    1 NOEUD(S) : N622
 LA VALEUR MAXIMALE DE DY      EST -4.30200662425417E-08 EN    1 NOEUD(S) : N559
 LA VALEUR MAXIMALE DE DZ      EST  4.11726555605663E-06 EN    1 NOEUD(S) : N564
 LA VALEUR MINIMALE DE DX      EST -6.08643978549605E-05 EN    1 NOEUD(S) : N527
 LA VALEUR MINIMALE DE DY      EST -5.23854834634229E-05 EN    1 NOEUD(S) : N457
```

```
 LA VALEUR MINIMALE DE DZ      EST -1.14312541871171E-04 EN    1 NOEUD(S) : N440

 ---------------------------------------------------------------------------
 ASTER 11.05.00 CONCEPT SOLU_4 CALCULE LE 15/06/2014 A 00:10:58 DE TYPE
EVOL_ELAS

              ENTITES TOPOLOGIQUES SELECTIONNEES
 GROUP_MA : Active_Surface

 ------>
 Min/Max displacement of Active_Surface

 LA VALEUR MAXIMALE DE DX      EST -1.42508970588132E-05 EN    1 NOEUD(S) : N1878
 LA VALEUR MAXIMALE DE DY      EST  4.95149182260603E-07 EN    1 NOEUD(S) : N559
 LA VALEUR MAXIMALE DE DZ      EST  2.92456828828774E-06 EN    1 NOEUD(S) : N564
 LA VALEUR MINIMALE DE DX      EST -4.71826010303129E-05 EN    1 NOEUD(S) : N469
 LA VALEUR MINIMALE DE DY      EST -4.29760022677855E-05 EN    1 NOEUD(S) : N457
 LA VALEUR MINIMALE DE DZ      EST -7.66389182468705E-05 EN    1 NOEUD(S) : N451
 <I> <FIN> FERMETURE DE LA BASE "GLOBALE" EFFECTUEE.

 <FIN> ArrÃªt normal dans "FIN".
 <I> <FIN> ARRET NORMAL DANS "FIN" PAR APPEL A "JEFINI".

 <I> <FIN> MEMOIRE JEVEUX MINIMALE REQUISE POUR L'EXECUTION :
30.63 Mo
 <I> <FIN> MEMOIRE JEVEUX OPTIMALE REQUISE POUR L'EXECUTION :
112.80 Mo
 <I> <FIN> MAXIMUM DE MEMOIRE UTILISEE PAR LE PROCESSUS LORS DE
L'EXECUTION :     709.77 Mo

 ********************************************************************************
 * COMMAND              :      USER :     SYSTEM :   USER+SYS :    ELAPSED *
 ********************************************************************************
 * init (jdc)           :      0.30 :      0.08 :      0.38 :      0.42 *
 *  . compile           :      0.00 :      0.00 :      0.00 :      0.01 *
 *  . exec_compile      :      0.11 :      0.04 :      0.15 :      0.13 *
 *  . report            :      0.01 :      0.00 :      0.01 :      0.03 *
 *  . build             :      0.00 :      0.00 :      0.00 :      0.00 *
 * DEBUT                :      0.05 :      0.04 :      0.09 :      0.12 *
 * DEFI_MATERIAU        :      0.01 :      0.00 :      0.01 :      0.02 *
 * DEFI_MATERIAU        :      0.00 :      0.00 :      0.00 :      0.01 *
 * LIRE_MAILLAGE        :      0.11 :      0.02 :      0.13 :      0.23 *
 * MODI_MAILLAGE        :      0.07 :      0.00 :      0.07 :      0.10 *
 * AFFE_MODELE          :      0.05 :      0.01 :      0.06 :      0.09 *
 * AFFE_MATERIAU        :      0.00 :      0.00 :      0.00 :      0.01 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.05 *
 * AFFE_CHAR_MECA       :      0.01 :      0.01 :      0.02 :      0.03 *
 * AFFE_CHAR_MECA       :      0.02 :      0.01 :      0.03 :      0.04 *
 * AFFE_CHAR_MECA       :      0.03 :      0.01 :      0.04 :      0.05 *
 * AFFE_CHAR_MECA       :      0.02 :      0.01 :      0.03 :      0.05 *
 * AFFE_CHAR_MECA       :      0.02 :      0.01 :      0.03 :      0.03 *
 * MECA_STATIQUE        :      0.84 :      0.16 :      1.00 :      1.01 *
 * CALC_CHAMP           :      0.45 :      0.08 :      0.53 :      0.53 *
```

```
* IMPR_RESU            :      0.03 :      0.01 :      0.04 :      0.06 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.01 *
* MECA_STATIQUE        :      0.84 :      0.17 :      1.01 :      1.04 *
* CALC_CHAMP           :      0.41 :      0.08 :      0.49 :      0.49 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.01 *
* IMPR_RESU            :      0.00 :      0.00 :      0.00 :      0.01 *
* MECA_STATIQUE        :      0.89 :      0.15 :      1.04 :      1.08 *
* CALC_CHAMP           :      0.42 :      0.08 :      0.50 :      0.48 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.01 *
* IMPR_RESU            :      0.00 :      0.00 :      0.00 :      0.01 *
* MECA_STATIQUE        :      0.86 :      0.17 :      1.03 :      1.05 *
* CALC_CHAMP           :      0.41 :      0.07 :      0.48 :      0.49 *
* IMPR_RESU            :      0.01 :      0.01 :      0.02 :      0.01 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.00 *
* MECA_STATIQUE        :      0.85 :      0.15 :      1.00 :      1.02 *
* CALC_CHAMP           :      0.44 :      0.07 :      0.51 :      0.52 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.02 *
* IMPR_RESU            :      0.01 :      0.00 :      0.01 :      0.00 *
* FIN                  :      0.03 :      0.10 :      0.13 :      0.30 *
*  . part Superviseur  :      0.39 :      0.13 :      0.52 :      0.75 *
*  . part Fortran      :      6.89 :      1.38 :      8.27 :      8.75 *
********************************************************************************
* TOTAL_JOB            :      7.29 :      1.51 :      8.80 :      9.50 *
********************************************************************************
```

## 10.12 FEA – Python scripts for studies

### 10.12.1 Winged_Flange_Setup1of3_Candidate2

```
import sys
import salome

salome.salome_init()
theStudy = salome.myStudy

import salome_notebook
notebook = salome_notebook.NoteBook(theStudy)
sys.path.insert( 0, r'/home/thomas/salome/Winged_Flange_Setup1of3/WF_Setup1of3_Candidate2')

###
### GEOM component
###

import GEOM
from salome.geom import geomBuilder
import math
import SALOMEDS


geompy = geomBuilder.New(theStudy)

O = geompy.MakeVertex(0, 0, 0)
OX = geompy.MakeVectorDXDYDZ(1, 0, 0)
OY = geompy.MakeVectorDXDYDZ(0, 1, 0)
OZ = geompy.MakeVectorDXDYDZ(0, 0, 1)
WF_solution_2_4parts_step_1 =
geompy.ImportSTEP("/home/thomas/salome/Winged_Flange_Setup1of3/WF_Setup1of3_
Candidate2/ Winged_Flange_Setup1of3_Candidate2.step")
[AnvilJaw,WP_0,WP,SlideJaw] =
geompy.ExtractShapes(Winged_Flange_Setup1of3_Candidate2_step_1,
geompy.ShapeType["SOLID"], True)
Partition_1 = geompy.MakePartition([AnvilJaw, WP, SlideJaw], [], [], [],
geompy.ShapeType["SOLID"], 0, [], 0, True)
geompy.addToStudy( O, 'O' )
geompy.addToStudy( OX, 'OX' )
geompy.addToStudy( OY, 'OY' )
geompy.addToStudy( OZ, 'OZ' )
geompy.addToStudy(Winged_Flange_Setup1of3_Candidate2_step_1, '
Winged_Flange_Setup1of3_Candidate2.step_1' )
geompy.addToStudyInFather(Winged_Flange_Setup1of3_Candidate2_step_1, AnvilJaw, 'AnvilJaw' )
geompy.addToStudyInFather(Winged_Flange_Setup1of3_Candidate2_step_1, WP, 'WP' )
geompy.addToStudyInFather( WF_solution_2_4parts_step_1, SlideJaw, 'SlideJaw' )
geompy.addToStudy( Partition_1, 'Partition_1' )
[AnvilJaw_1, WP_1, SlideJaw_1] = geompy.RestoreGivenSubShapes(Partition_1, [AnvilJaw, WP,
SlideJaw], GEOM.FSM_GetInPlaceByHistory, False, False)
WP_edge_113 = geompy.GetSubShape(WP_1, [113])
Vertex_1 = geompy.MakeVertexOnCurveByCoord(WP_edge_113, 0.4395, 0.45, 0.507)
WP_vertex_118 = geompy.GetSubShape(WP_1, [118])
```

```
Line_1 = geompy.MakeLineTwoPnt(Vertex_1, WP_vertex_118)
Partition_2 = geompy.MakePartition([Line_1, AnvilJaw_1, WP_1, SlideJaw_1], [], [], [],
geompy.ShapeType["SOLID"], 0, [], 0, True)
geompy.addToStudyInFather( WP_1, WP_edge_113, 'WP:edge_113' )
geompy.addToStudy( Vertex_1, 'Vertex_1' )
geompy.addToStudyInFather( WP_1, WP_vertex_118, 'WP:vertex_118' )
geompy.addToStudy( Line_1, 'Line_1' )
geompy.addToStudy( Partition_2, 'Partition_2' )
[geomObj_1, AnvilJaw_2, WP_2, geomObj_2, geomObj_3, SlideJaw_2] =
geompy.RestoreGivenSubShapes(Partition_2, [Line_1, AnvilJaw_1, WP_1, WP_edge_113,
WP_vertex_118, SlideJaw_1], GEOM.FSM_GetInPlaceByHistory, False, False)
Anv_Back = geompy.CreateGroup(AnvilJaw_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Anv_Back, [127])
Anv_Cut_Fil = geompy.CreateGroup(AnvilJaw_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Anv_Cut_Fil, [133])
Active_Surface = geompy.CreateGroup(WP_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Active_Surface, [34])
Sl_Back = geompy.CreateGroup(SlideJaw_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Sl_Back, [72])
Feature_0 = geompy.CreateGroup(WP_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Feature_0, [34])
Feature_1 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_1, [16])
Feature_2 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_2, [54, 51])
Feature_3 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_3, [46, 49])
Feature_4 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_4, [64, 61])
Feature_5 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_5, [56, 59])
geompy.addToStudyInFather( AnvilJaw_2, Anv_Back, 'Anv_Back' )
geompy.addToStudyInFather( WP_2, Feature_0, 'Feature_0' )
geompy.addToStudyInFather( WP_2, Feature_1, 'Feature_1' )
geompy.addToStudyInFather( WP_2, Feature_2, 'Feature_2' )
geompy.addToStudyInFather( AnvilJaw_2, Anv_Cut_Fil, 'Anv_Cut_Fil' )
geompy.addToStudyInFather( WP_2, Active_Surface, 'Active_Surface' )
geompy.addToStudyInFather( SlideJaw_2, Sl_Back, 'Sl_Back' )
geompy.addToStudyInFather( WP_2, Feature_3, 'Feature_3' )
geompy.addToStudyInFather( WP_2, Feature_4, 'Feature_4' )
geompy.addToStudyInFather( WP_2, Feature_5, 'Feature_5' )

###
### SMESH component
###

import  SMESH, SALOMEDS
from salome.smesh import smeshBuilder

smesh = smeshBuilder.New(theStudy)
Mesh_Partition = smesh.Mesh(Partition_2)
Regular_1D = Mesh_Partition.Segment()
Max_Size_1 = Regular_1D.MaxSize(0.01)
```

```
MEFISTO_2D = Mesh_Partition.Triangle(algo=smeshBuilder.MEFISTO)
NETGEN_3D = Mesh_Partition.Tetrahedron()
Max_Size_2 = smesh.CreateHypothesis('MaxLength')
Max_Size_2.SetLength( 0.003 )
status = Mesh_Partition.AddHypothesis(Regular_1D,WP_2)
status = Mesh_Partition.AddHypothesis(Max_Size_2,WP_2)
status = Mesh_Partition.AddHypothesis(MEFISTO_2D,WP_2)
status = Mesh_Partition.AddHypothesis(NETGEN_3D,WP_2)
Max_Size_3 = smesh.CreateHypothesis('MaxLength')
Max_Size_3.SetLength( 0.0005 )
status = Mesh_Partition.AddHypothesis(Regular_1D,Anv_Cut_Fil)
status = Mesh_Partition.AddHypothesis(Max_Size_3,Anv_Cut_Fil)
status = Mesh_Partition.AddHypothesis(MEFISTO_2D,Anv_Cut_Fil)
isDone = Mesh_Partition.Compute()
AnvilJaw_3 = Mesh_Partition.GroupOnGeom(AnvilJaw_2,'AnvilJaw',SMESH.VOLUME)
AnvilJaw_4 = Mesh_Partition.GroupOnGeom(AnvilJaw_2,'AnvilJaw',SMESH.NODE)
WP_3 = Mesh_Partition.GroupOnGeom(WP_2,'WP',SMESH.VOLUME)
WP_4 = Mesh_Partition.GroupOnGeom(WP_2,'WP',SMESH.NODE)
SlideJaw_3 = Mesh_Partition.GroupOnGeom(SlideJaw_2,'SlideJaw',SMESH.VOLUME)
SlideJaw_4 = Mesh_Partition.GroupOnGeom(SlideJaw_2,'SlideJaw',SMESH.NODE)
Anv_Back_1 = Mesh_Partition.GroupOnGeom(Anv_Back,'Anv_Back',SMESH.FACE)
Active_Surface_1 = Mesh_Partition.GroupOnGeom(Active_Surface,'Active_Surface',SMESH.FACE)
Sl_Back_1 = Mesh_Partition.GroupOnGeom(Sl_Back,'Sl_Back',SMESH.FACE)
Anv_Back_2 = Mesh_Partition.GroupOnGeom(Anv_Back,'Anv_Back',SMESH.NODE)
Active_Surface_2 =
Mesh_Partition.GroupOnGeom(Active_Surface,'Active_Surface',SMESH.NODE)
Sl_Back_2 = Mesh_Partition.GroupOnGeom(Sl_Back,'Sl_Back',SMESH.NODE)
Feature_0_1 = Mesh_Partition.GroupOnGeom(Feature_0,'Feature_0',SMESH.FACE)
Feature_1_1 = Mesh_Partition.GroupOnGeom(Feature_1,'Feature_1',SMESH.EDGE)
Feature_2_1 = Mesh_Partition.GroupOnGeom(Feature_2,'Feature_2',SMESH.EDGE)
Feature_3_1 = Mesh_Partition.GroupOnGeom(Feature_3,'Feature_3',SMESH.EDGE)
Feature_4_1 = Mesh_Partition.GroupOnGeom(Feature_4,'Feature_4',SMESH.EDGE)
Feature_5_1 = Mesh_Partition.GroupOnGeom(Feature_5,'Feature_5',SMESH.EDGE)
Feature_0_2 = Mesh_Partition.GroupOnGeom(Feature_0,'Feature_0',SMESH.NODE)
Feature_1_2 = Mesh_Partition.GroupOnGeom(Feature_1,'Feature_1',SMESH.NODE)
Feature_2_2 = Mesh_Partition.GroupOnGeom(Feature_2,'Feature_2',SMESH.NODE)
Feature_3_2 = Mesh_Partition.GroupOnGeom(Feature_3,'Feature_3',SMESH.NODE)
Feature_4_2 = Mesh_Partition.GroupOnGeom(Feature_4,'Feature_4',SMESH.NODE)
Feature_5_2 = Mesh_Partition.GroupOnGeom(Feature_5,'Feature_5',SMESH.NODE)
smesh.SetName(Mesh_Partition, 'Mesh_Partition')
WP_5 = Mesh_Partition.GetSubMesh( WP_2, 'WP' )
Cut_Fil = Mesh_Partition.GetSubMesh( Anv_Cut_Fil, 'SubMesh_1' )
Mesh_Partition.ExportMED(
r'/home/thomas/salome/Winged_Flange_Setup1of3/WF_Setup1of3_Candidate2/Winged_Flange_1of3
_2_Partition.med', 0, SMESH.MED_V2_2, 1, None ,1)


## Set names of Mesh objects
smesh.SetName(Regular_1D.GetAlgorithm(), 'Regular_1D')
smesh.SetName(NETGEN_3D.GetAlgorithm(), 'NETGEN_3D')
smesh.SetName(MEFISTO_2D.GetAlgorithm(), 'MEFISTO_2D')
smesh.SetName(Max_Size_2, 'Max Size_2')
smesh.SetName(Max_Size_1, 'Max Size_1')
```

```
smesh.SetName(Feature_5_2, 'Feature_5')
smesh.SetName(Feature_4_2, 'Feature_4')
smesh.SetName(Feature_3_2, 'Feature_3')
smesh.SetName(Anv_Back_1, 'Anv_Back')
smesh.SetName(Max_Size_3, 'Max Size_3')
smesh.SetName(Active_Surface_1, 'Active_Surface')
smesh.SetName(Sl_Back_1, 'Sl_Back')
smesh.SetName(Feature_0_1, 'Feature_0')
smesh.SetName(Cut_Fil, 'Cut_Fil')
smesh.SetName(Mesh_Partition.GetMesh(), 'Mesh_Partition')
smesh.SetName(SlideJaw_3, 'SlideJaw')
smesh.SetName(WP_3, 'WP')
smesh.SetName(AnvilJaw_3, 'AnvilJaw')
smesh.SetName(Feature_1_2, 'Feature_1')
smesh.SetName(Feature_2_2, 'Feature_2')
smesh.SetName(Feature_1_1, 'Feature_1')
smesh.SetName(WP_5, 'WP')
smesh.SetName(Feature_3_1, 'Feature_3')
smesh.SetName(Feature_2_1, 'Feature_2')
smesh.SetName(Feature_5_1, 'Feature_5')
smesh.SetName(WP_4, 'WP')
smesh.SetName(Feature_4_1, 'Feature_4')
smesh.SetName(SlideJaw_4, 'SlideJaw')
smesh.SetName(AnvilJaw_4, 'AnvilJaw')
smesh.SetName(Sl_Back_2, 'Sl_Back')
smesh.SetName(Feature_0_2, 'Feature_0')
smesh.SetName(Anv_Back_2, 'Anv_Back')
smesh.SetName(Active_Surface_2, 'Active_Surface')


###
```

### 10.12.2 Cyl_D60_H15_Setup1of1_Candidate3

```
import sys
import salome

salome.salome_init()
theStudy = salome.myStudy

import salome_notebook
notebook = salome_notebook.NoteBook(theStudy)
sys.path.insert( 0, r'/home/thomas/salome/Cyl_D60_H15')

###
### GEOM component
###

import GEOM
from salome.geom import geomBuilder
import math
import SALOMEDS


geompy = geomBuilder.New(theStudy)

O = geompy.MakeVertex(0, 0, 0)
OX = geompy.MakeVectorDXDYDZ(1, 0, 0)
OY = geompy.MakeVectorDXDYDZ(0, 1, 0)
OZ = geompy.MakeVectorDXDYDZ(0, 0, 1)
Cyl_D60_H15_Setup1of1_Assembly3_synth_step_1 =
geompy.ImportSTEP("/home/thomas/salome/Cyl_D60_H15/Cyl_D60_H15_Setup1of1_Assembly3_s
ynth.step")
[AnvilJaw,Cylinder_D60_H15_0_WP,WP,SlideJaw] =
geompy.ExtractShapes(Cyl_D60_H15_Setup1of1_Assembly3_step_1,
geompy.ShapeType["SOLID"], True)
Partition_1 = geompy.MakePartition([AnvilJaw, WP, SlideJaw], [], [], [],
geompy.ShapeType["SOLID"], 0, [], 1, True)
geompy.addToStudy( O, 'O' )
geompy.addToStudy( OX, 'OX' )
geompy.addToStudy( OY, 'OY' )
geompy.addToStudy( OZ, 'OZ' )
geompy.addToStudy( Cyl_D60_H15_Setup1of1_Assembly3_step_1,
'Cyl_D60_H15_Setup1of1_Assembly3.step_1' )
geompy.addToStudyInFather( Cyl_D60_H15_Setup1of1_Assembly3_step_1, AnvilJaw, 'AnvilJaw' )
geompy.addToStudyInFather( Cyl_D60_H15_Setup1of1_Assembly3_step_1, WP, 'WP' )
geompy.addToStudyInFather( Cyl_D60_H15_Setup1of1_Assembly3_step_1, SlideJaw, 'SlideJaw' )
geompy.addToStudy( Partition_1, 'Partition_1' )
[AnvilJaw_1, WP_1, SlideJaw_1] = geompy.RestoreGivenSubShapes(Partition_1, [AnvilJaw, WP,
SlideJaw], GEOM.FSM_GetInPlaceByHistory, False, False)
Vertex_1 = geompy.MakeVertex(1, 1, 1.01)
WP_vertex_96 = geompy.GetSubShape(WP_2, [96])
WP_vertex_93 = geompy.GetSubShape(WP_2, [93])
Arc_1 = geompy.MakeArcCenter(Vertex_1, WP_vertex_96, WP_vertex_93,False)
WP_vertex_112 = geompy.GetSubShape(WP_2, [112])
```

```
WP_vertex_109 = geompy.GetSubShape(WP_2, [109])
Vertex_2 = geompy.MakeVertex(1, 1, 1.006)
Arc_2 = geompy.MakeArcCenter(Vertex_2, WP_vertex_112, WP_vertex_109,False)
Partition_2 = geompy.MakePartition([Arc_1, Arc_2, AnvilJaw_2, WP_2, SlideJaw_2], [], [], [],
geompy.ShapeType["SOLID"], 0, [], 0, True)
geompy.addToStudy( Vertex_1, 'Vertex_1' )
geompy.addToStudyInFather( WP_2, WP_vertex_96, 'WP:vertex_96' )
geompy.addToStudyInFather( WP_2, WP_vertex_93, 'WP:vertex_93' )
geompy.addToStudy( Arc_1, 'Arc_1' )
geompy.addToStudyInFather( WP_2, WP_vertex_112, 'WP:vertex_112' )
geompy.addToStudyInFather( WP_2, WP_vertex_109, 'WP:vertex_109' )
geompy.addToStudy( Vertex_2, 'Vertex_2' )
geompy.addToStudy( Partition_2, 'Partition_2' )
geompy.addToStudy( Arc_2, 'Arc_2' )
[geomObj_1, AnvilJaw_2, WP_2, geomObj_2, geomObj_3, SlideJaw_2] =
geompy.RestoreGivenSubShapes(Partition_2, [Arc_1, Arc_2, AnvilJaw_1, WP_1, WP_vertex_96,
WP_vertex_93, WP_vertex_112, WP_vertex_109, SlideJaw_1], GEOM.FSM_GetInPlaceByHistory,
False, False)
Anv_Back = geompy.CreateGroup(AnvilJaw_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Anv_Back, [3])
Sl_Back = geompy.CreateGroup(SlideJaw_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Sl_Back, [3])
Active_Surface = geompy.CreateGroup(WP_2, geompy.ShapeType["FACE"])
geompy.UnionIDs(Active_Surface, [41])
Feature_0 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_0, [40])
Feature_1 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_1, [26])
Feature_2 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_2, [95])
Feature_3 = geompy.CreateGroup(WP_2, geompy.ShapeType["EDGE"])
geompy.UnionIDs(Feature_3, [70, 63])
geompy.addToStudyInFather( AnvilJaw_2, Anv_Back, 'Anv_Back' )
geompy.addToStudyInFather( WP_2, Feature_0, 'Feature_0' )
geompy.addToStudyInFather( WP_2, Feature_1, 'Feature_1' )
geompy.addToStudyInFather( WP_2, Feature_2, 'Feature_2' )
geompy.addToStudyInFather( AnvilJaw_2, Anv_Cut_Fil, 'Anv_Cut_Fil' )
geompy.addToStudyInFather( WP_2, Active_Surface, 'Active_Surface' )
geompy.addToStudyInFather( SlideJaw_2, Sl_Back, 'Sl_Back' )
geompy.addToStudyInFather( WP_2, Feature_3, 'Feature_3' )
geompy.addToStudyInFather( WP_2, Feature_4, 'Feature_4' )
geompy.addToStudyInFather( WP_2, Feature_5, 'Feature_5' )

###
### SMESH component
###

import  SMESH, SALOMEDS
from salome.smesh import smeshBuilder

smesh = smeshBuilder.New(theStudy)
Partition_Cyl_D60_H15 = smesh.Mesh(Partition_2)
Regular_1D = Partition_Cyl_D60_H15.Segment()
```

```
Max_Size_1 = Regular_1D.MaxSize(0.01)
MEFISTO_2D = Partition_Cyl_D60_H15.Triangle(algo=smeshBuilder.MEFISTO)
NETGEN_3D = Partition_Cyl_D60_H15.Tetrahedron()
Max_Size_2 = smesh.CreateHypothesis('MaxLength')
Max_Size_2.SetLength( 0.003 )
status = Partition_Cyl_D60_H15.AddHypothesis(Regular_1D,WP_2)
status = Partition_Cyl_D60_H15.AddHypothesis(Max_Size_2,WP_2)
status = Partition_Cyl_D60_H15.AddHypothesis(MEFISTO_2D,WP_2)
status = Partition_Cyl_D60_H15.AddHypothesis(NETGEN_3D,WP_2)
isDone = Partition_Cyl_D60_H15.Compute()
AnvilJaw_3 = Partition_Cyl_D60_H15.GroupOnGeom(AnvilJaw_2,'AnvilJaw',SMESH.VOLUME)
AnvilJaw_4 = Partition_Cyl_D60_H15.GroupOnGeom(AnvilJaw_2,'AnvilJaw',SMESH.NODE)
WP_3 = Partition_Cyl_D60_H15.GroupOnGeom(WP_2,'WP',SMESH.VOLUME)
WP_4 = Partition_Cyl_D60_H15.GroupOnGeom(WP_2,'WP',SMESH.NODE)
SlideJaw_3 = Partition_Cyl_D60_H15.GroupOnGeom(SlideJaw_2,'SlideJaw',SMESH.VOLUME)
SlideJaw_4 = Partition_Cyl_D60_H15.GroupOnGeom(SlideJaw_2,'SlideJaw',SMESH.NODE)
Anv_Back_1 = Partition_Cyl_D60_H15.GroupOnGeom(Anv_Back,'Anv_Back',SMESH.FACE)
Sl_Back_1 = Partition_Cyl_D60_H15.GroupOnGeom(Sl_Back,'Sl_Back',SMESH.NODE)
Active_Surface_1 =
Partition_Cyl_D60_H15.GroupOnGeom(Active_Surface,'Active_Surface',SMESH.FACE)
Feature_0_1 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_0,'Feature_0',SMESH.EDGE)
Feature_1_1 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_1,'Feature_1',SMESH.EDGE)
Feature_2_1 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_2,'Feature_2',SMESH.EDGE)
Feature_3_1 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_3,'Feature_3',SMESH.EDGE)
Anv_Back_2 = Partition_Cyl_D60_H15.GroupOnGeom(Anv_Back,'Anv_Back',SMESH.NODE)
Sl_Back_2 = Partition_Cyl_D60_H15.GroupOnGeom(Sl_Back,'Sl_Back',SMESH.NODE)
Active_Surface_2 =
Partition_Cyl_D60_H15.GroupOnGeom(Active_Surface,'Active_Surface',SMESH.NODE)
Feature_0_2 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_0,'Feature_0',SMESH.NODE)
Feature_1_2 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_1,'Feature_1',SMESH.NODE)
Feature_2_2 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_2,'Feature_2',SMESH.NODE)
Feature_3_2 = Partition_Cyl_D60_H15.GroupOnGeom(Feature_3,'Feature_3',SMESH.NODE)
Feature_3_Mom1 = Partition_Cyl_D60_H15.CreateEmptyGroup( SMESH.NODE, 'Feature_3_Mom1'
)
nbAdd = Feature_3_Mom1.Add( [ 471 ] )
Feature_3_Mom1.SetColor( SALOMEDS.Color( 1, 0.666667, 0 ))
Feature_3_Mom2 = Partition_Cyl_D60_H15.CreateEmptyGroup( SMESH.NODE, 'Feature_3_Mom2'
)
nbAdd = Feature_3_Mom2.Add( [ 467 ] )
Feature_3_Mom2.SetColor( SALOMEDS.Color( 1, 0.666667, 0 ))

smesh.SetName(Partition_Cyl_D60_H15, 'Partition_Cyl_D60_H15')
WP_5 = Mesh_Partition.GetSubMesh( WP_2, 'WP' )
Partition_Cyl_D60_H15.ExportMED(
r'/home/thomas/salome/Cyl_D60_H15/Partition_Cyl_D60_H15.med', 0, SMESH.MED_V2_2, 1,
None ,1)

## Set names of Mesh objects
smesh.SetName(Regular_1D.GetAlgorithm(), 'Regular_1D')
smesh.SetName(NETGEN_3D.GetAlgorithm(), 'NETGEN_3D')
smesh.SetName(MEFISTO_2D.GetAlgorithm(), 'MEFISTO_2D')
smesh.SetName(Max_Size_2, 'Max Size_2')
smesh.SetName(Max_Size_1, 'Max Size_1')
```

```
smesh.SetName(Feature_3_2, 'Feature_3')
smesh.SetName(Anv_Back_1, 'Anv_Back')
smesh.SetName(Active_Surface_1, 'Active_Surface')
smesh.SetName(Sl_Back_1, 'Sl_Back')
smesh.SetName(Feature_0_1, 'Feature_0')
smesh.SetName(Partition_Cyl_D60_H15.GetMesh(), 'Partition_Cyl_D60_H15')
smesh.SetName(SlideJaw_3, 'SlideJaw')
smesh.SetName(WP_3, 'WP')
smesh.SetName(AnvilJaw_3, 'AnvilJaw')
smesh.SetName(Feature_1_2, 'Feature_1')
smesh.SetName(Feature_2_2, 'Feature_2')
smesh.SetName(Feature_1_1, 'Feature_1')
smesh.SetName(WP_5, 'WP')
smesh.SetName(Feature_3_1, 'Feature_3')
smesh.SetName(Feature_2_1, 'Feature_2')
smesh.SetName(WP_4, 'WP')
smesh.SetName(SlideJaw_4, 'SlideJaw')
smesh.SetName(AnvilJaw_4, 'AnvilJaw')
smesh.SetName(Sl_Back_2, 'Sl_Back')
smesh.SetName(Feature_0_2, 'Feature_0')
smesh.SetName(Anv_Back_2, 'Anv_Back')
smesh.SetName(Active_Surface_2, 'Active_Surface')

###
```

## 10.13 FEA - EFICAS scripts for studies

### 10.13.1 Winged_Flange_Setup1of3

```
DEBUT();

# Define user variables
# SI units: m, N
# E-modulus and Poissons ratio of workpiece material
E_WP=68900000000.0;
NU_WP=0.33;

# Load/Force settings - set sign such that force acts in direction wanted w.r.t the global COS
# Clamping force in negative x-direction
F_CLAMP=3480; # Forces in N

# Machining forces for each feature of the setup
# All forces in absolute values, directions are applied in load settings

F_CUT_FEAT_0=1525;
F_FEED_FEAT_0=1525;
Active_Surface_Area=0.013; # for WF_Setup1of3 in square meters
F_CUT_0=F_CUT_FEAT_0/Active_Surface_Area;
F_FEED_0=F_FEED_FEAT_0/Active_Surface_Area;

F_CUT_FEAT_1=1740;
F_FEED_FEAT_1=1740;
Radius_Tool_Feat_1=0.005; # in meter
Radius_Feat_1=0.016;
F_CUT_1=F_CUT_FEAT_1*Radius_Feat_1/(pow(Radius_Tool_Feat_1,2)*pi);
F_FEED_1=F_FEED_FEAT_1*Radius_Feat_1/(pow(Radius_Tool_Feat_1,2)*pi);

F_CUT_FEAT_2=1281;
F_FEED_FEAT_2=266;
NUMBER_NODES_FEAT_2=8; # This can be retrieved via the node group using Python
(MAIL_PY)
F_CUT_2=F_CUT_FEAT_2/4; # F_CUT applied as two nodal forces to represent momentum
F_FEED_2=F_FEED_FEAT_2/NUMBER_NODES_FEAT_2; #F_FEED split onto the nodes of the
drill hole circumference

# Define material of jaws (6061 T6 aluminum)
MAT_JAWS=DEFI_MATERIAU(ELAS=_F(E=68900000000.0,
                    NU=0.33,),);

# Define material of workpiece
MAT_WP=DEFI_MATERIAU(ELAS=_F(E=E_WP,
                    NU=NU_WP,),);

# Define mesh type
MAIL=LIRE_MAILLAGE(FORMAT='MED',);

# Define outward normal of group of element faces for loading, s.t. pressure is correctly applied
MAIL=MODI_MAILLAGE(reuse=MAIL,
```

```
            MAILLAGE=MAIL,
            ORIE_PEAU_3D=_F(GROUP_MA=('Sl_Back','Anv_Back','Active_Surface',),),);

# Set calculation mode
# Instead of TOUT='OUI' specific mesh groups can be used: AFFE=_F(GROUP_MA=('xxx','yyy',),
FEMLin=AFFE_MODELE(MAILLAGE=MAIL,
            AFFE=_F(TOUT='OUI',
                PHENOMENE='MECANIQUE',
                MODELISATION='3D',),);

# Assign material to mesh
# Different materials can be assigend to mesh groups by AFFE=_F(GROUP_MA=('xxx',)
MATE=AFFE_MATERIAU(MAILLAGE=MAIL,
                AFFE=(_F(GROUP_MA=('AnvilJaw','SlideJaw',),MATER=MAT_JAWS,),
                    _F(GROUP_MA='WP',MATER=MAT_WP,),),);

# Apply displacement boundary conditions that are constant for all calculations
BC=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FACE_IMPO=(_F(GROUP_MA='Sl_Back',DY=0.0,DZ=0.0,),
                                _F(GROUP_MA='Anv_Back',DX=0.0,DY=0.0,DZ=0.0),),);

count=[0,1,2,3,4,5,6];
LOAD=[None]*len(count);
SOLU=[None]*len(count);

# Add clamping and machining loads per feature
# To work against locators feed direction is in neg. x-direction and cutting firce in neg. y-direction
(right-turning tool)
LOAD[0]=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),);

LOAD[1]=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FORCE_FACE=(_F(GROUP_MA='Active_Surface',FY=-F_CUT_0,FX=-
F_FEED_0,),
                                _F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),);

LOAD[2]=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                        FORCE_ARETE=(_F(GROUP_MA='Feature_1',FY=-F_CUT_1,FX=-
F_FEED_1,),),);

LOAD[3]=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                        FORCE_NODALE=(_F(GROUP_NO='Feature_2',FZ=F_FEED_2,),
                                _F(GROUP_NO='Feature_2_Mom1',FX=F_CUT_2,),
                    _F(GROUP_NO='Feature_2_Mom2',FX=-F_CUT_2,),),);

LOAD[4]=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                        FORCE_NODALE=(_F(GROUP_NO='Feature_3',FZ=F_FEED_2,),
                                _F(GROUP_NO='Feature_3_Mom1',FX=F_CUT_2,),
                    _F(GROUP_NO='Feature_3_Mom2',FX=-F_CUT_2,),),);
```

```
LOAD[5]=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                        FORCE_NODALE=(_F(GROUP_NO='Feature_4',FZ=F_FEED_2,),
                                      _F(GROUP_NO='Feature_4_Mom1',FX=F_CUT_2,),
                    _F(GROUP_NO='Feature_4_Mom2',FX=-F_CUT_2,),),);

LOAD[6]=AFFE_CHAR_MECA(MODELE=FEMLin,
                        FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                        FORCE_NODALE=(_F(GROUP_NO='Feature_5',FZ=F_FEED_2,),
                                      _F(GROUP_NO='Feature_5_Mom1',FX=F_CUT_2,),
                    _F(GROUP_NO='Feature_5_Mom2',FX=-F_CUT_2,),),);



for counter in count:

        # Define linear MUMPS solver
        SOLU[counter]=MECA_STATIQUE(MODELE=FEMLin,
                CHAM_MATER=MATE,
                EXCIT=(_F(CHARGE=BC,),
                     _F(CHARGE=LOAD[counter],),),
                SOLVEUR=_F(METHODE='MUMPS',),);

        # Calculate results
        SOLU[counter]=CALC_CHAMP(reuse=SOLU[counter],
                RESULTAT=SOLU[counter],
                CONTRAINTE=('SIGM_NOEU','SIGM_ELNO',),
                DEFORMATION='EPSI_NOEU',
                );

        # Write results to med file
        IMPR_RESU(FORMAT='MED',
                UNITE=80,
                RESU=_F(RESULTAT=SOLU[counter],
                NOM_CHAM=('DEPL',)));

        # Write min/max values of WP node displacement to table
        IMPR_RESU(FORMAT='RESULTAT',
                RESU=_F(RESULTAT=SOLU[counter],
                NOM_CHAM='DEPL',
                FORM_TABL='OUI',
                GROUP_MA='Active_Surface',
                VALE_MAX='OUI',
                VALE_MIN='OUI',
                SOUS_TITRE='Min/Max displacement of Active_Surface',),);

FIN();
```

### 10.13.2 Cyl_D60_H15_Setup1of1

```
DEBUT();

# Define user variables
# SI units: m, N
# E-modulus and Poissons ratio of workpiece material
E_WP=68900000000.0;
NU_WP=0.33;

# Load/Force settings - set sign such that force acts in direction wanted w.r.t the global COS
# Clamping force in negative x-direction
F_CLAMP=4066; # Forces in N as calculated with safety factor 2.0

# Define the counter for the looped calculation, 0 for clamp force only then one more for every feature
count=[0,1,2,3,4];
LOAD=[None]*len(count);
SOLU=[None]*len(count);

# Machining force settings for each feature of the setup
# Milling cutting force in neg. x-direction, causes feeding force in pos. y-direction (right-turning tool)

#Circular pocket milling feature 0, applied as edge load
F_CUT_FEAT_0=1740;
F_FEED_FEAT_0=1740;
Radius_Tool_Feat_0=0.005; # in meter
Radius_Feat_0=0.027;
F_CUT_0=F_CUT_FEAT_0*Radius_Feat_0/(pow(Radius_Tool_Feat_0,2)*pi); # Force of half tool
circumference w.r.t. feature line length (half feature circumference)
F_FEED_0=F_FEED_FEAT_0*Radius_Feat_0/(pow(Radius_Tool_Feat_0,2)*pi);

#Groove milling feature 1 and feature 2, applied as edge load
F_CUT_FEAT_1=1509;
F_FEED_FEAT_1=1509;
Radius_Tool_Feat_1=0.003; # in meter
Radius_Feat_1=0.003;
F_CUT_1=F_CUT_FEAT_1*Radius_Feat_1/(pow(Radius_Tool_Feat_1,2)*pi); # Force of half tool
circumference w.r.t. feature line length (half feature circumference)
F_FEED_1=F_FEED_FEAT_1*Radius_Feat_1/(pow(Radius_Tool_Feat_1,2)*pi);

#Drilling feature 3, applied as nodal loads
F_CUT_FEAT_3=2033;
F_FEED_FEAT_3=924;
NUMBER_NODES_FEAT_3=10; # This can be retrieved via the node group using Python
(MAIL_PY)
F_CUT_3=F_CUT_FEAT_3/2; # F_CUT applied as two nodal forces in opposite direction to
represent momentum
F_FEED_3=F_FEED_FEAT_3/NUMBER_NODES_FEAT_3; #F_FEED split onto the nodes of the
drill hole circumference
# Define material of jaws (6061 T6 aluminum)
MAT_JAWS=DEFI_MATERIAU(ELAS=_F(E=68900000000.0,
                    NU=0.33,),);
```

```
# Define material of workpiece
MAT_WP=DEFI_MATERIAU(ELAS=_F(E=E_WP,
                NU=NU_WP,),);

# Define mesh type
MAIL=LIRE_MAILLAGE(FORMAT='MED',);

# Define outward normal of group of element faces for loading, s.t. pressure is correctly applied
MAIL=MODI_MAILLAGE(reuse=MAIL,
            MAILLAGE=MAIL,
            ORIE_PEAU_3D=_F(GROUP_MA=('Sl_Back','Anv_Back','Active_Surface',),),);

# Set calculation mode
# Instead of TOUT='OUI' specific mesh groups can be used: AFFE=_F(GROUP_MA=('xxx','yyy',),
FEMLin=AFFE_MODELE(MAILLAGE=MAIL,
            AFFE=_F(TOUT='OUI',
                PHENOMENE='MECANIQUE',
                MODELISATION='3D',),);

# Assign material to mesh
MATE=AFFE_MATERIAU(MAILLAGE=MAIL,
                    AFFE=(_F(GROUP_MA=('AnvilJaw','SlideJaw',),MATER=MAT_JAWS,),
                          _F(GROUP_MA='WP',MATER=MAT_WP,),),);

# Apply displacement boundary conditions that are constant for all calculations
BC=AFFE_CHAR_MECA(MODELE=FEMLin,
                          FACE_IMPO=(_F(GROUP_MA='Sl_Back',DY=0.0,DZ=0.0,),
                                     _F(GROUP_MA='Anv_Back',DX=0.0,DY=0.0,DZ=0.0),),);

# Add clamping and machining loads per feature
LOAD[0]=AFFE_CHAR_MECA(MODELE=FEMLin,
                          FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-
F_CLAMP/0.004,),),);

LOAD[1]=AFFE_CHAR_MECA(MODELE=FEMLin,
                          FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                          FORCE_ARETE=(_F(GROUP_MA='Feature_1',FX=-F_FEED_0,FY=-
F_CUT_0,),),);

LOAD[2]=AFFE_CHAR_MECA(MODELE=FEMLin,
                          FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                          FORCE_ARETE=(_F(GROUP_MA='Feature_2',FX=-F_CUT_1,FY=-
F_FEED_1,),),);

LOAD[3]=AFFE_CHAR_MECA(MODELE=FEMLin,
                          FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                          FORCE_ARETE=(_F(GROUP_MA='Feature_3',FX=-F_CUT_1,FY=-
F_FEED_1,),),);

LOAD[4]=AFFE_CHAR_MECA(MODELE=FEMLin,
                          FORCE_FACE=(_F(GROUP_MA='Sl_Back',FX=-F_CLAMP/0.004,),),
                          FORCE_NODALE=(_F(GROUP_NO='Feature_4',FZ=F_FEED_3,),
                                        _F(GROUP_NO='Feature_4_Mom1',FX=F_CUT_3,),
```

```
                                                    _F(GROUP_NO='Feature_4_Mom2',FX=-
F_CUT_3,),),);

for counter in count:

        # Define linear MUMPS solver
        SOLU[counter]=MECA_STATIQUE(MODELE=FEMLin,
                CHAM_MATER=MATE,
                EXCIT=(_F(CHARGE=BC,),
                     _F(CHARGE=LOAD[counter],),),
                SOLVEUR=_F(METHODE='MUMPS',),);

        # Calculate results
        SOLU[counter]=CALC_CHAMP(reuse=SOLU[counter],
                RESULTAT=SOLU[counter],
                CONTRAINTE=('SIGM_NOEU','SIGM_ELNO',),
                DEFORMATION='EPSI_NOEU',
                );

        # Write results to med file
        IMPR_RESU(FORMAT='MED',
                UNITE=80,
                RESU=_F(RESULTAT=SOLU[counter],
                NOM_CHAM=('DEPL',)));

        # Write min/max values of WP node displacement to table
        IMPR_RESU(FORMAT='RESULTAT',
                RESU=_F(RESULTAT=SOLU[counter],
                NOM_CHAM='DEPL',
                FORM_TABL='OUI',
                GROUP_MA='Active_Surface',
                VALE_MAX='OUI',
                VALE_MIN='OUI',
                SOUS_TITRE='Min/Max displacement of Active_Surface',),);

FIN();
```