A Rule-based Collaborative Modelling System for Infrastructure Design

Flurl M., Singer D., Mundani R.-P., Rank E., Borrmann A.
Leonhard Obermeyer Center, Technische Universität München, Germany
flurl@bv.tum.de, dominic.singer@tum.de

Abstract. Within the frame of the German Research Foundation project "3DTracks", a collaboration platform has been designed to enable different planers to work synchronously on a shared geometry model, thereby using Computer-Aided Design (CAD) tools they are accustomed to. The basis for this modelling process is a newly developed procedural model that supports the typical feature operations of modern CAD tools and especially the idea of geometrical and dimensional constraints. Since there are many situations in which these geometric and dimensional constraints are not sufficient to express engineering boundary conditions, there is a need for more complex rules in order to reflect the design intent of the modelling engineer. Thus, we investigate possibilities for the integration of adequate rules into the procedural model and into the modelling process, respectively. In this paper, we present our strategy to integrate these collaborative rules, delineate how these rules are constructed, and finally give some examples of their application and the resulting benefit. We thus extend the well-known approaches of rule-based functionalities of CAD-systems to interoperable models shared in a synchronous collaboration environment.

1. Introduction

In recent years, several researchers have intensively studied the process of collaborative geometric modelling. In the course of these studies, they found that procedural geometry models can serve as a suitable basis for collaborative modelling scenarios. In simplified terms, these procedural models store the models' individual construction steps as well as the dependencies between them (Pratt 2003). In particular, these dependencies comprise the idea of constraints to explicitly describe relations between different geometry features such as parallelism of lines, concentricity of circles, specific symmetries etc. (Pratt & Junhwan 2005, Ma et al. 2008). By applying these constraints, parts of the design intent of the model creators (i.e. engineers) are incorporated into the procedural geometry description (Bianconi 2006, Ma et al. 2008) – in contrast to explicit geometry descriptions that merely account for the resulting geometry, e.g. vertices, edges, and faces.

The research project "3DTracks"[1]has brought forth new methodologies to improve collaborative work in the field of synchronous geometry design and semantic modelling processes. One of the main fields of research in this respect focused on 3D modelling techniques to support the design of alignment-based infrastructure projects concerning the modelling of subway tunnels. This new methodology comprises two main aspects: the novel multi-scale procedural geometry model itself as well as the usage of this model in a collaborative synchronous modelling process (Borrmann et al. 2014). In particular, a dedicated collaboration system – following a client and server architecture – has been developed. It enables several modelling experts to work on a single geometry model simultaneously, using the CAD tools they are accustomed to. At the same time, the collaboration server ensures the consistency of the shared model and its local replicas residing on the client sides (Flurl et al. 2014).

[1] http://www.3dtracks.kit.edu/english/overview.php

The newly developed multi-scale procedural geometry model supports the well-established modelling features such as extrusions, sweeps, Boolean operations, and 2D-sketches as a common basis for many civil engineering constructions. A sketch itself supports the aforementioned constraint mechanisms (Jubierre & Borrmann 2013). In general, geometrical constraints describe relations between selected geometry entities, while dimensional constraints allow the user to define specific dimensions of given elements, such as the length of a specific line or the angle between two different lines (Brüderlin & Roller 1998).

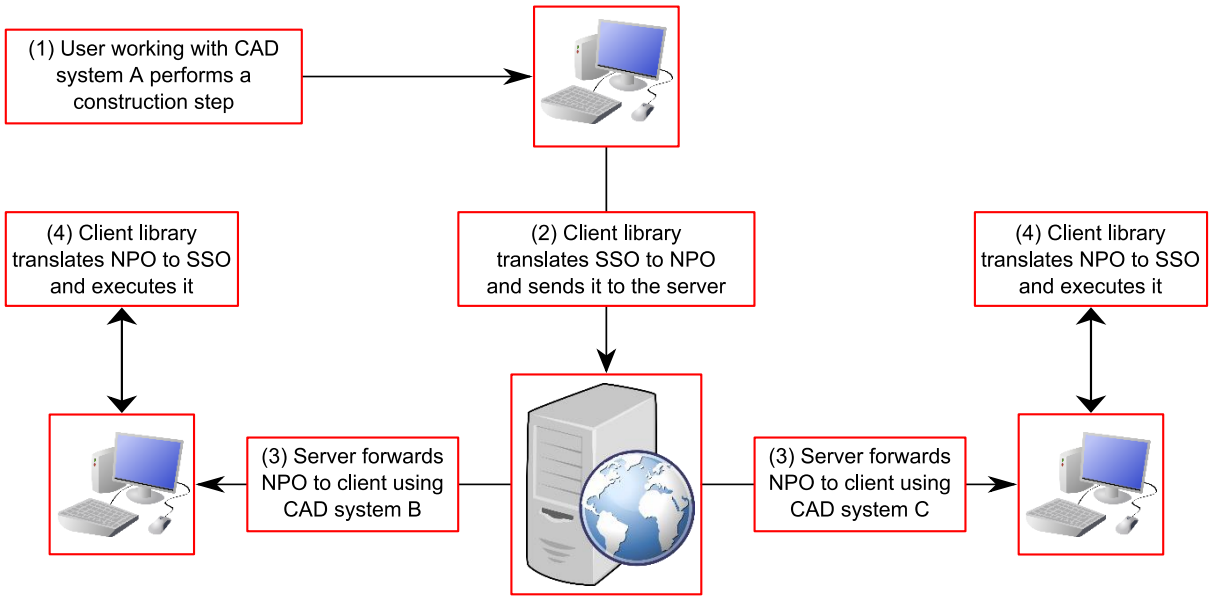

Figure 1: Basic workflow using the new procedural model in a collaborative modelling session

The main idea in order to use this new procedural model in a collaborative modelling process is that *Neutral Procedural Operations* (NPO) are translated into *System Specific Operations* (SSO) of the several distinguished CAD tools (Figure 1). This translation process is managed by libraries that are integrated into the local CAD tools via their APIs. Prototypically, we developed libraries for the three CAD systems Autodesk Inventor[2], Siemens NX[3] and the in-house geometry framework TUM.GeoFrame (Sorger et al. 2014).

2. Problem Definition

Though geometrical and dimensional constraints are able to describe many facets of engineering design intent, they do not cover all aspects. Thus, constraints appear to be insufficient even for rules based on simple inequalities. If we consider, for example, a simple upper limit of a given distance, we might encounter the following situation: A simple tunnel tube is described by the Boolean Operation *subtract* using two sweep operations based on a spline defining the principle track course and two sketches defining the inner and the out tunnel hull (Figure 2).

Obviously, this *subtract* operation is only possible as long as the circle defining the outer tunnel hull has a radius that is larger than the radius of the circle defining the inner tunnel hull.

[2] http://www.autodesk.de/products/inventor/overview
[3] http://www.plm.automation.siemens.com/de_de/products/nx/

Although this problem seems to be quite simple, it is not possible to describe the condition $r_2 + 0.2 < r_1$ in common CAD tools, since these usually only support equations.

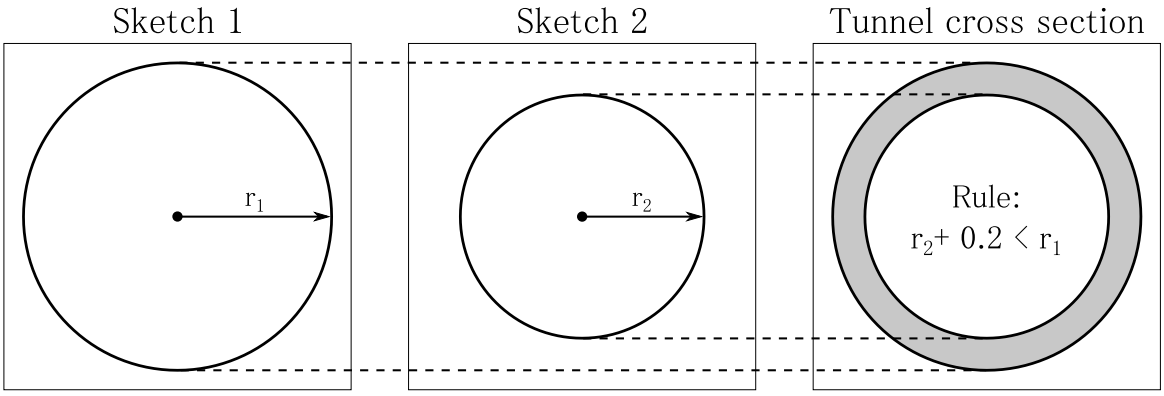

Figure 2: Simple tunnel tube based on two sketches defining the tunnel profile

Rule-based systems, on the other hand, are well known for capturing and considering complex design rules (Reijnders, A.W. 2012). Today, modern commercial CAD tools integrate proprietary rule-based engineering systems, since these systems are still single-user applications or – in the best case – provide active collaborative features solely for CAD tools of the same product line. Hence, users in a collaborative modelling process are not free to use the modelling tool of their choice. Moreover, the capturing, processing and reuse of engineering intent is still dependent on proprietary CAD systems.

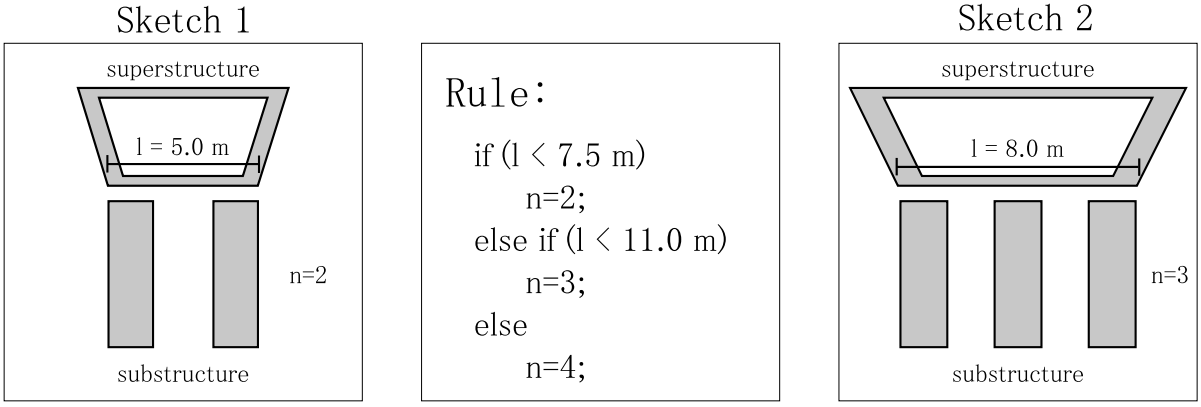

Figure 3: Simple bridge model, where the number of pillars depends on the bottom width of the superstructure.

As an example for a more complex rule, the number of pillars used in the construction of a bridge may depend on the width, given by a specific parameter of the superstructure (Figure 3). In the geometric modelling process, a sketch defines this superstructure, while a dimensional constraint referencing a specific parameter determines the width. As soon as this parameter exceeds a specific length, the numbers of pillars should change from two to three or three to four, and vice versa if the parameter falls below a certain length. In a modelling system, the sketch defining the basis for the pillars should be adapted to the new situation automatically.

Though this situation seems quite simple, it is not possible to define this kind of rules in a collaborative workflow, in particular in a collaborative workflow using different CAD modelling tools.

3. Rule-based Systems

Rule-based systems originate from the field of Artificial Intelligence in the 1970s (Hayes-Roth & Jacobstein, N. 1994; Dym 1985). Nowadays, rule-based systems are applied in many different areas and knowledge domains such as diagnostics systems, production systems, business rules management systems, decision support systems, and knowledge-based engineering. As a matter of principle, a rule-based system consists of the components of a knowledge base, a rule (or inference) engine, and a user interface. Via the user interface, which could also be embedded in a CAD system, a user can either add rules or query the system. (Griffin, Lewis 1989). The rule engine (see Figure 4) matches the rules stored in the knowledge base against some facts, in our case the procedural model (*Pattern Matching*). All rules that need to be processed are called conflicts. Before execution, these conflicts are listed and ordered into an agenda using a conflict solving strategy. It is also possible to include some forward or backward chaining techniques. The main advantage of using a rule engine is the separation of rule definition (declarative) and the execution of consequences.

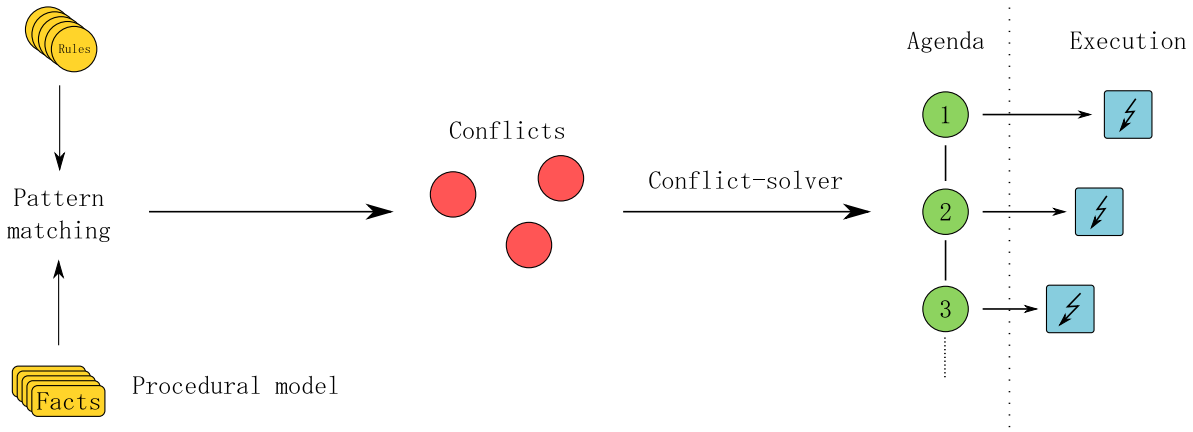

Figure 4: Rule engine

Today, most proprietary CAD software tools feature a rule-based system, such as the modules iLogic or Engineer-To-Order (ETO) for Autodesk Inventor and the software product Knowledge Fusion that provides rule-based functionalities for Siemens NX. However, these tools are still supposed to be used in interaction with their proprietary parent software systems such as Inventor or Siemens NX. In the scope of this work, we want to overcome those boundaries by merging rule-based services into a manufacturer-independent collaboration platform. We thus extend the well-known approaches of rule-based functionality of CAD-systems to interoperable models shared in a synchronous collaboration environment.

4. Solution

To resolve issues outlined above, we present a strategy to integrate complex rules into the "3DTracks" collaboration platform. In addition to the possibility of using the well-known constraints, we allow the user to define rules that determine relations between different geometrical elements in a more general way than it can be done drawing on constraints. The user

uploads and stores these rules in the collaboration platform in order to incorporate them into the shared procedural model. Once the rule is activated and one of the users modifies a specific procedural element that affects this specific rule, a *rule engine* evaluates the change and induces the execution of predefined steps according to the rules definition and the outcome of the evaluation process.

Form of Collaborative Rules

In general, the simplest form of a rule is

IF *antecedent* THEN *consequent*,

where *antecedent* represents a condition and c*onsequent* defines steps that one has to undertake if the evaluation of the *antecedent* results in the value *true* (Reijnders 2012). The first simple extension in this definition is to allow an alternative, i.e.

IF *antecedent* THEN *consequent*_1 ELSE *consequent*_2,

additionally allowing the definition of steps to be executed if the evaluation of the *antecedent* process results in the value *false*. The next extension is to allow the usage of conditional "IF … THEN … ELSE" statements in the consequents statement. This leads to a recursive definition of a rule in a very general form. This form of rule is supported in our approach. Before we provide a strictly formal definition at the end of this section, we would like to focus on the connection between rule definition and the geometry elements as well as the definition of possible consequences.

The Conditions in the *Antecedents* Part

In our case, a condition included in the *antecedent* part of a rule consists of algebraic relations between parameters defining the shape of geometrical elements or construction steps resulting in geometry elements, respectively. In modern modelling tools, the parametric definition of certain values is of fundamental relevance. Mostly, these tools are feature-based, whereby features are usually defined as construction steps representing a high-level geometry operation *that makes it possible for a shape designer to avoid having to work from the low level or individual curve and surface elements* (Mun 2003). In particular, these high-level construction steps as well as the basic sketch operations are driven by the aforementioned parameters that define certain sizes – the height of an extrusion, the length of a line, or the diameter of a circle in a sketch definition, for example (see Figure 5).

In our definition of rules, the *antecedent* comprises arbitrary algebraic combinations of dimensions or parameters, respectively, that result in a valid equation or inequality. The syntactic validity is verified by a parser during the definition, while the semantic validity lies in the engineer's responsibility.

The procedural model schema determines which parameters are allowed to be combined, e.g. the radius of a circle is defined as "*RuleUsable"*. The specific parameter in a rule definition is given by a description in the manner of an object-oriented programming language, e.g. $Sketches["SketchId\_02"].Circles["CircleId\_01"].Radius$ describes the radius of the circle with the id "$CircleId\_01$" from the sketch with the id "$SketchId\_02$". For example, the *antecedent* of the rule mentioned in the introduction can be delineated for a specific application

$$Sketches["SketchId\_02"].Circles["CircleId\_01"].Radius + 0.2 < Sketches["SketchId\_03"].Circles["CircleId\_01"].Radius$$

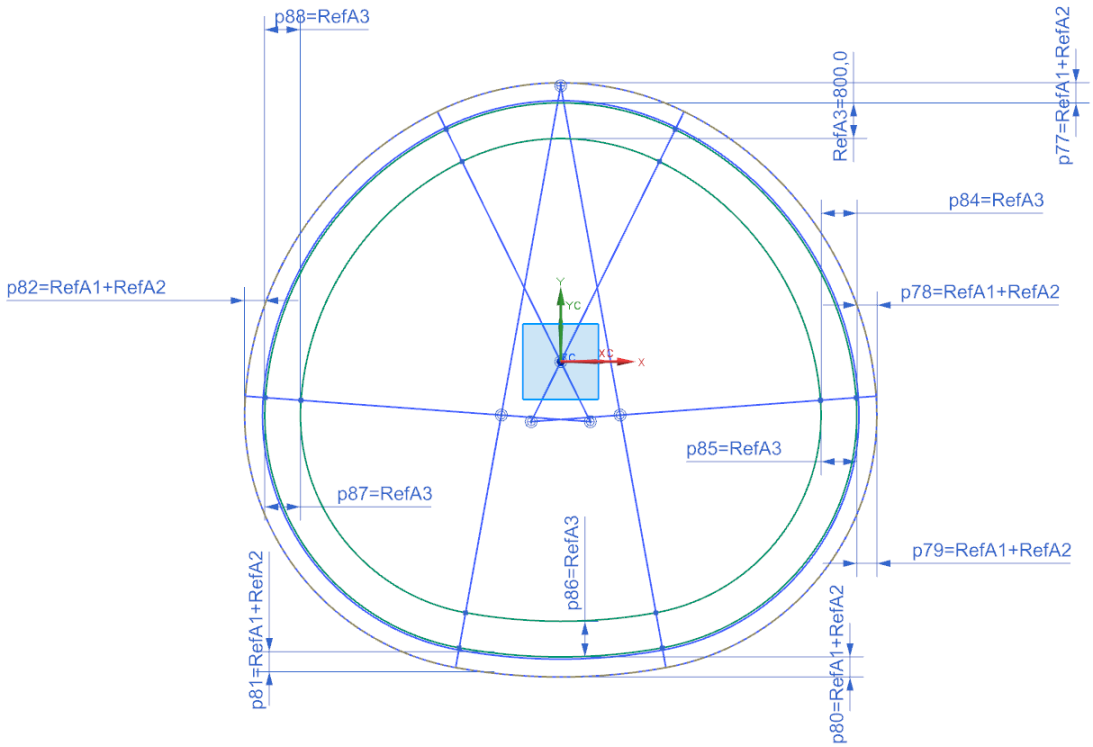

Figure 5: A parametrical sketch describing the cross section of a subway tunnel

For the users, the definition concerning the procedural model schema is transparent, since the editor for the defining rules (see chapter 5) only offers parameters of which the property "*RuleUsable*" is set to *true*.

The Execution Steps in the *Consequents* Part

The *consequent* statement defines steps to be executed after the *antecedent* of the rule has been evaluated to *true*. In a collaborative system, this statement has to be interpreted by the collaboration server, initiating specific geometric operations to be performed by the affected clients. The statements themselves may contain other conditions (resulting from the recursive rule definition) that have to be evaluated. We distinguish three different types of possible execution steps: warnings given to the user, rejection of a model modification step, and finally, a modifying mechanism in order to manipulate the shared model or to resolve a problem state. To fully understand this, one has to reconsider the process of a modification step in the given collaborative scenario.

As soon as a modelling expert starts the modification of an element, the collaboration server – automatically notified by the client application – locks this element for the other participating users (Flurl et al. 2014). The user finishes his modification and sends the modified operation to the collaboration server, which then incorporates the modified operation into the central shared model and forwards the modification to the other participating clients, implicitly forcing them to integrate this modification immediately. The collaboration server unlocks the specific operation and finally notifies the editing user about the success of his modification step.

In the rule-based collaborative system, before incorporating the modified operation into the shared model, all rules referencing this modified operation are re-evaluated. To facilitate this, after adding a rule, a reference to this rule is automatically added to the "*ReferencedRule*"-collection of all operations concerned. This collection is also used if an element is deleted:

referenced rules are deactivated and the user has to redefine the obviously incorrect rule version.

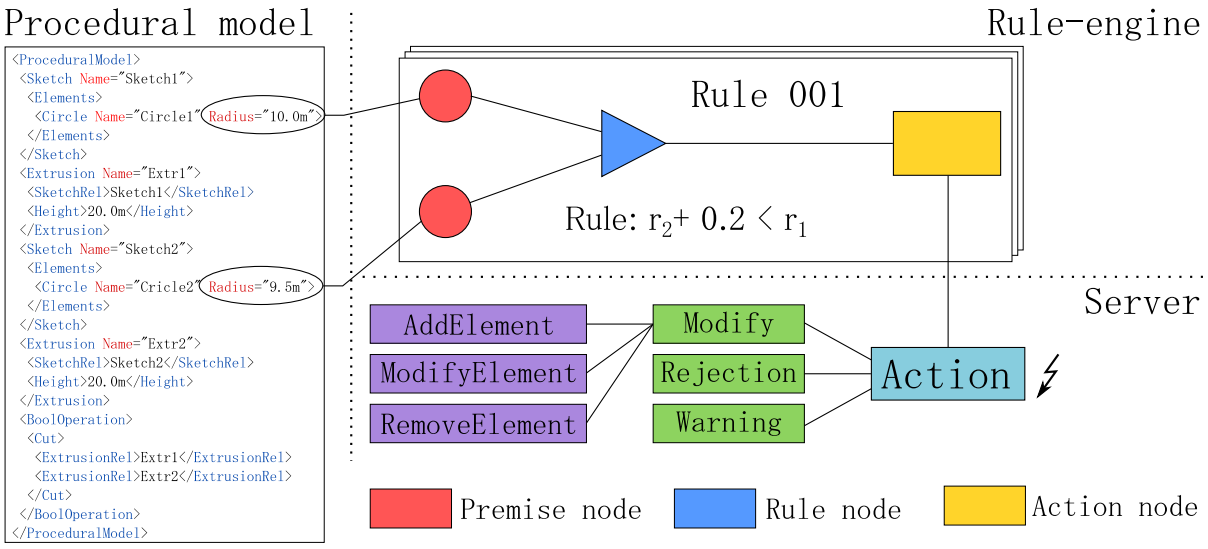

Figure 6: Collaboration Server and rule engine interaction

In the following paragraphs, we will briefly discuss the three fundamental cases of consequents that were mentioned before. Since no action is required if the evaluation of a rule results in the value *false*, our focus here lies on cases resulting in the value *true*. Then, the simplest option is that the *consequent* part of a rule is set to "*Reject*" (Figure 6). In this case, the modified operation is not incorporated into the shared model, while the user is forced to undo the modification in his local copy of the shared model. A similarly simple option is that the *consequent* part of a rule is set to "*Warning*" (Figure 6). In this case, the operation is modified as described above (see also Figure 1), but a warning message is sent to the user, relying on the user's ability to resolve possible inconsistencies or emerging problems, respectively. This is a typical strategy in collaborative systems (Munson 1998).

The last case "*ModifyModel*" is more complex. Here, one can deposit "any possible" set of operations to be executed onto the shared procedural model on the server side. This set of operations comprises three different types of operations: Existing construction steps or geometrical elements can be modified by (1) changing specific parameters, (2) adding new construction steps and (3) by removing existing ones. These three types of operations are executed using the existing interface provided by the collaboration server, including the ability to automatically lock a set of elements in order to avoid inconsistencies. In particular, if at least one of the referenced elements cannot be locked – indicating that another user is working on the element in question – the modification is rejected, as in the first case described above. In addition, the server automatically forces all the other participating users to include the various modifications steps locally. A simple example for this type of rule in the context of the previous example might be

DEFINE $param\_ref1$:= $Sketches$[SketchId_02].$Circles$[CircleId_01].$Radius$;

DEFINE $param\_ref2$:= $Sketches$[SketchId_03].$Circles$[CircleId_01].$Radius$;

IF $param\_ref1 > param\_ref2$ THEN

$$\text{MODIFYELEMENT}(\text{"SketchId\_02"}, param\_ref1, param\_ref2 - 0.5) \qquad (4.1)$$

Within this example, the radius of the sketch “SketchId_02” is automatically adapted when it exceeds a valid upper limit given in the antecedent. Defining *param_ref1* and *param_ref2* strongly simplifies the readability of the rule.

Following these first explanatory examples, we now formally define a rule in the following according to Figure 7.

```
Ident           = Letter {Letter | Digit}
Sign            = +|-
Number          = Sign Digit {Digit}
Digit           = 0|1|2|3|4|5|6|7|8|9
Letter          = A|B|C ... Z

Rule            = Rule Ident { MainStat }

MainStat        = if ( Expr ) Stat [else Stat]

Stat            = if ( Expr ) Stat [else Stat]
                | Reject | Warning | ModifyModel

ModifyModel     = {ModEle | AddEle | RemEle}
ModEle          = Modify(ProcIdent,ParamIdent,Number|SimpleExpr)

Expr            = SimpleExpr RelOp SimpleExpr
SimpleExpr      = Term  {AddOp Term}
Term            = Factor {MulOp Factor}
Factor          = ProcIdent.ParamIdent | Number

AddOp           = +|-
MulOp           = *|/
RelOp           ==|<|>|<=|>=

ProcIdent       = Ident
ParamIdent      = Ident
```

Figure 7: Formal definition of a rule

Combining Different Rules and Conflicting Situations

In general, different rules may lead to conflicting situations – for example, if a second rule to the rule is added to the rule above (4.1):

IF $param\_ref1 < param\_ref2$ THEN

$$\text{MODIFYELEMENT}(\text{"SketchId\_02"}, param\_ref1, param\_ref2 - 1.5)$$

the final result of the evaluation of the different rules depends on the order in which the given rules are processed. There are several strategies to resolve conflicts – in particular strategies that are automatically provided by the rule engines (Chan et. al. 1985, Hicks 2007). In the current version of our approach, we follow the very simple idea of processing the rules by their given order of definition, thus relying on the users’ ability to resolve conflicts. In future work, we will incorporate more elaborate conflict-resolving-strategies.

5. Implementation

In order to simplify the process of defining rules, we prototypically implemented a simple rule editor (see Figure 8), giving the user a possibility to select the involved geometrical elements by simply picking them from the GUI while defining the rules.

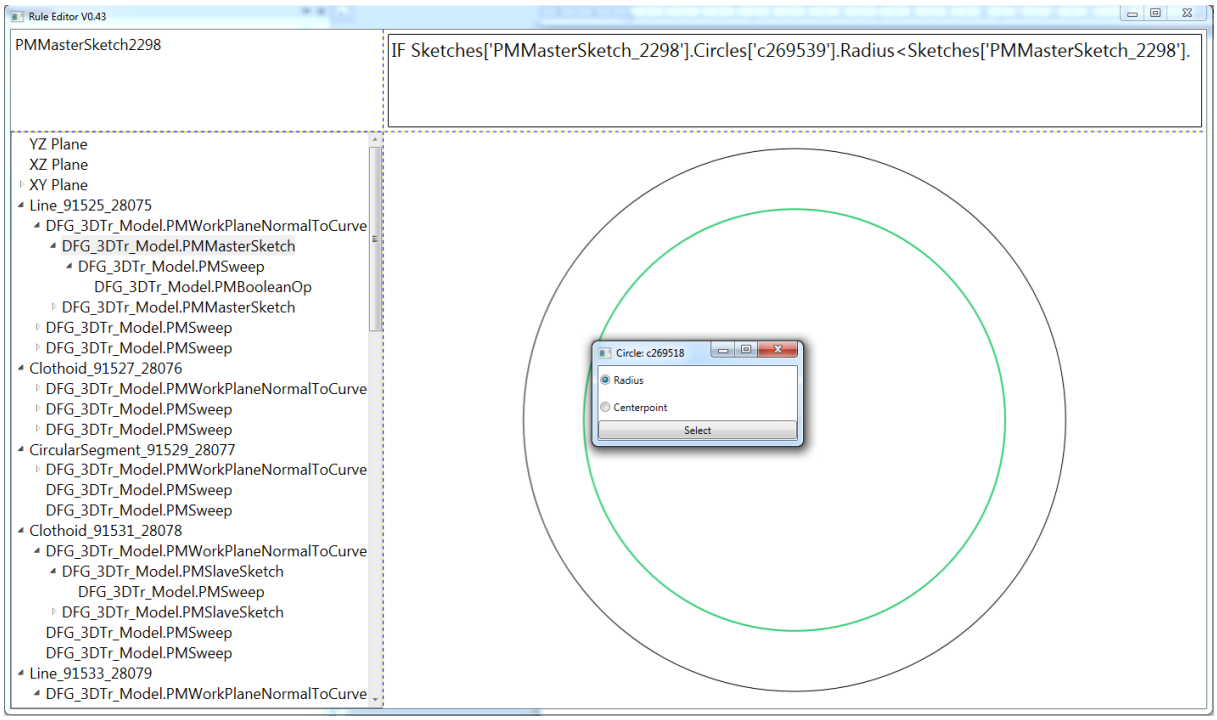

Figure 8: The rule editor supports the user during the task of defining rules

Thereby, a feature tree allows the user to choose the operations that are referenced by a specific rule. After selecting an operation, the associated parameters are presented in a popup selection panel and can thus be defined easily. This way, operations and parameters are gradually combined to an equation or inequality representing the antecedent of the specified rule.

6. Conclusion and Outlook

In the field of procedural geometric modelling, the concept of constraints is well established, yet often not fully sufficient to represent the designers' intent. We presented an approach to mitigate this problem in the field of collaborative CAD modelling. It could be shown that our approach is suitable for integrating more complex rules into a collaborative modelling process. The implementation is accomplished by a collaboration platform developed within the German Research Foundation project "3DTracks". We delineated the formal syntax of a set of rules to determine structure of antecedent and consequent. While these rules can be defined dynamically in our software prototype, we validated the usability of the collaborative rule-based system using a simple and intuitive example that is nonetheless representative for an extensive field of similar problems.

In future work, we will investigate and integrate more elaborate conflict-solving strategies and, in particular, revise our role system for users. Additionally, we are planning to extend our modelling environment in order to enable the integration of actual knowledge-based engineering functionalities and techniques.

Acknowledgement:
The research presented in this paper has been financially supported by the German Research Foundation (DFG) within the frame of the 3DTracks research unit. This is gratefully acknowledged.

References

Bianconi, F.: Towards a Procedural CAD Model for Data exchange: problems and perspective. In Proc. Congreso Internacional Conjunto XVII Ingegraf–XV ADM: De la Tradición al Futuro, 2005

Borrmann, A.; Flurl, M.; Jubierre, J.R.; Mundani, R.-P.; Rank, E.: Synchronous collaborative tunnel design based on consistency-preserving multi-scale models. Advanced Engineering Informatics, 2014

Borrmann, A.; Ji, Y.; Jubierre, J. R.; Flurl M.: Procedural Modeling: A new approach to multi-scale design in infrastructure projects, In: Proc. of the EG-ICE Workshop on Intelligent Computing in Engineering, Herrsching, Germany, 2012

Brüderlin, B., Roller, D.: Geometric Constraint Solving and Applications, Springer; 1998

Chan, S. J.; Yu, P. L.: A dynamic conflict resolution model: Acceptability of alternatives and conflict solvability. In: J Optim Theory Appl 4v7 (3), pp. 269–284. DOI: 10.1007/BF00941494., 1985

Dym, Clive L.: EXPERT SYSTEMS: New approaches to computer-aided engineering. In: Engineering with Computers 1 (1), S. 9–25. DOI: 10.1007/BF01200335., 1985

Flurl, M.; Mundani, R.-P.; Rank, E.: Graph-based Concurrency Control for Multi-Scale Procedural Models. In: Proc. of the 10th European Conference on Product & Process Modeling, Vienna, Austria, 2014

Griffin, N. L.; Lewis, F. D.: A rule-based inference engine which is optimal and VLSI implementable. In: IEEE International Workshop on Tools for Artificial Intelligence. Fairfax, VA, USA, 23-25 Oct., S. 246–251., 1989

Hayes-Roth, F.; Jacobstein, N.: The state of knowledge-based systems. In: Commun. ACM 37 (3), pp. 26–39. DOI: 10.1145/175247.175249., 1994

Hicks, Richard C.: The no inference engine theory — Performing conflict resolution during development. In: Decision Support Systems 43 (2), pp. 435–444. DOI: 10.1016/j.dss.2006.11.001., 2007

Jubierre, J. R.; Borrmann, A.: Cross-submodel consistency preservation in multi-scale engineering models. In: Proc. of the 14th International Conference on Civil, Structural and Environmental Engineering Computing (CSC2013), Cagliari, Sardinia, Italy, 2013

La Rocca, G.: Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. In: Advanced Engineering Informatics 26 (2), pp. 159–179. DOI: 10.1016/j.aei.2012.02.002, 2012

Ma, Y.-S., Chen, G., Thimm, G.: Paradigm shift: unified and associative feature-based concurrent and collaborative engineering, Springer, 2008

Mun, D., Hana, S., Kima, J., Ohb, Y.: A set of standard modeling commands for the history-based parametric approach. In: Computer-Aided Design 35, pp. 1171–1179, 2003

Munson J., Dewan, P.: A Concurrency Control Framework for Collaborative Systems. In: Proc of the 1996 ACM conference on Computer supported cooperative work, Boston, USA, 1996

Pratt, M.: Procedural Modelling, Generative Geometry, and the International Standard ISO 10303 (STEP). In: Mathematics of Surfaces, Springer, pp. 320-337, 2003

Pratt, M., Junhwan, K.: Experience in the Exchange of Procedural Shape Models using ISO 10303 (STEP). In Proc. of the 2006 ACM symposium on Solid and physical modeling, Cardiff, UK, 2006

Reijnders, A.W.: Integrating Knowledge Management and Knowledge-Based Engineering. Formal and Platform Independent Representation of Engineering Rules, TU Delft, Delft, 2012

Sorger, C.; Frischmann, F.; Kollmannsberger, S.; Rank, E.: TUM.GeoFrame: Automated high-order hexahedral mesh generation for shell-like structures, Engineering with Computers 30 (1), pp. 41-56, 2014