TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Luftfahrtsysteme

Methodik für den modellbasierten Vorentwurf von Flugzeugsystemen

Benedikt Andreas Mohr

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

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Florian Holzapfel
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Mirko Hornung
	2.	Univ.-Prof. Dr.-Ing. Horst Baier

Die Dissertation wurde am 29.02.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 29.06.2012 angenommen.

VORWORT

Die vorliegende Arbeit entstand während meiner Zeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Luftfahrtsysteme der Technischen Universität München. Der Inhalt der Dissertation wurde aus den Erkenntnissen abgeleitet, die ich im Rahmen der Durchführung verschiedener Forschungsprojekte im universitären und industriellen Umfeld gewinnen durfte.

Zunächst möchte ich meinem Doktorvater Prof. Dr.-Ing. Mirko Hornung besonderen Dank aussprechen, da er stets großes Vertrauen in mich gesetzt hat und mir neben den notwendigen Freiräumen in der Ausgestaltung der Arbeit wertvolle inhaltliche Unterstützung gab. Ebenso möchte ich Prof. Dr.-Ing. Horst Baier für die Prüfung der Dissertation und die stets kollegiale Zusammenarbeit danken. Herrn Prof. Dr.-Ing. Florian Holzapfel bin ich für die Übernahme des Vorsitzes im Promotionsausschuss dankbar. An dieser Stelle möchte ich einen besonderen Dank an Dr.-Ing. Wolfgang Schmidt richten, der mir zu Beginn meiner Arbeit am Lehrstuhl große Unterstützung zukommen ließ und mich zur Forschung an meinem Thema motivierte.

Die Arbeit am Lehrstuhl war durch die enge Freundschaft zu meinen Kollegen Bastian Figlar und Joachim Schömann sehr erfreulich und produktiv. Sie haben mich maßgeblich in den erlebten Höhen und Tiefen unterstützt und durch ihre hilfreichen Kommentare sehr großen Anteil an der erfolgreichen Ausarbeitung dieser Dissertation beigetragen. Weiterhin bedanke ich mich bei Stephan Eelman, Tilman Richter, Kay Plötner, Peter Phleps, Philipp Böck, Gerald Öttl, Christian Rößler, Daniel Paulus und Sebastian Speck für ihre Unterstützung und das stets kollegiale Arbeitsklima.

Meinen Eltern danke ich für die Möglichkeiten, die sie mir im Leben eröffnet haben und die bedingungslose Unterstützung meines Lebensweges. Nicht zuletzt danke ich aus ganzem Herzen meiner Freundin Johanna, die immer für mich da war und mich stets ermutigt hat meinen Weg zu gehen – wohin er auch führen mag.

München, August 2012 Benedikt Mohr

KURZFASSUNG

Die Entwicklung von Flugzeugsystemen ist geprägt durch steigende Anforderungen an Funktionsweise und Qualität was eine höhere Effektivität und Effizienz der zugrundeliegenden Entwicklungsprozesse bedingt. Durch die inhärente Komplexität luftfahrttechnischer Produkte besteht die Entwicklung aus einer Vielzahl anspruchsvoller Aktivitäten, die ein koordiniertes Zusammenwirken vieler Spezialisten verschiedener Fachdisziplinen erfordern. Die domänenspezifischen Tätigkeiten im Entwicklungsprozess werden heutzutage durch ausgereifte Modellierungswerkzeuge unterstützt, die bereits die effektive Nutzung digitaler Modelle erlauben und damit Zeit- und Kostenvorteile gegenüber realen Modellen generieren. Durch die enorme Vielzahl an Softwarewerkzeugen hat sich eine komplexe und heterogene IT-Landschaft etabliert, die eine Effizienzsteigerung von Prozessen aufgrund von Kommunikations- und Koordinationsdefiziten stark einschränkt.

Vor diesem Hintergrund diskutiert diese Arbeit Ansätze zur Integration der heterogenen Softwarelandschaft in der Luftfahrtindustrie unter Berücksichtigung der aktuellen Produkt-, Prozess- und Organisationsstrukturen. Als Lösungsansatz wird die Erstellung und Implementierung eines zentralen Systemmodells identifiziert, welches im Entwicklungsprozess anfallende Systeminformationen in einheitlicher und übersichtlicher Form erfasst. Das Systemmodell dient als zentraler Informationsspeicher und erlaubt es den beteiligten Entwicklungsdomänen, Daten über Domänengrenzen hinweg und zwischen Werkzeugen auszutauschen. Darüber hinaus dient das Systemmodell als Grundlage zur Erstellung von fachspezifischen Sichtweisen auf die Systemstruktur sowie zur Parallelisierung und Automatisierung von Entwicklungsabläufen. Weiterhin propagiert diese Arbeit die Modellierung von Systemanforderungen und -funktionen sowie den zugrundeliegenden Organisationsstrukturen zur Schaffung einer integrierten Entwicklungsumgebung.

Die Anwendbarkeit der Methode sowie das Potenzial zur Unterstützung und Verbesserung aktueller Systementwicklungsprozesse in der Luftfahrt werden abschließend an einem industrienahen Anwendungsbeispiel demonstriert, bevor die Arbeit mit einer Zusammenfassung der erreichten Ergebnisse und einem Ausblick für folgende Untersuchungen schließt.

ABSTRACT

The development of aircraft systems is characterized by a rising demand for higher functionality and quality which results in the demand for more efficient system development processes. The intrinsic complexity of aerospace products makes their development a coordinated effort of demanding activities of various specialists from different technical domains. The domain specific activities are today very well supported by dedicated tools that allow for efficient use of digital models and thus a reduction of time and cost compared to physical modeling. However, the enormous diversity of tools in current development processes creates a highly complex and heterogeneous software infrastructure that constrains the augmentation of process efficiency due to its immanent communication and coordination deficits.

Against this background this thesis discusses different approaches for the integration of heterogeneous software tools within the aerospace industry taking the product-, process- and organizational structures into account. The development and implementation of a system model, which is capable of capturing system life cycle data in a clear and common style, is identified as a possible solution and therefore further developed. The system model serves as a central data memory medium and allows for the exchange of data sets across domain boundaries as well as among software tools. Furthermore, the system model enables the creation of subject-specific views on the system structure as well as the parallelization and automation of development activities. In addition, the thesis proposes the modeling of system requirements, functions and the organizational structure to enable an integrated development environment.

The practicality of the proposed method as well as the potential to enhance current aerospace system development processes is demonstrated with an industry-oriented use case. The thesis concludes with a résumé on the achieved results and an outlook for possible future work.

TERMINOLOGIE

Bauteil – Ein Bauteil stellt einen nicht zerlegbaren Grundbaustein von Systemen und Produkten dar.

Komponente – Eine Komponente stellt einen Zusammenschluss von mehreren Bauteilen dar, die eine physische Einheit bilden.

Modul – Ein Modul ist eine unabhängige Komponente, die eine bestimmte Produktfunktion erfüllt.

Element – Der Begriff eines Elements erweitert die Betrachtung von materiellen Bauteilen um den Aspekt immaterieller Bestandteile wie Software.

Produkt – Ein Produkt ist ein künstliches Erzeugnis, das einem bestimmten, vordefinierten Zweck erfüllt (LINDEMANN, 2011). Demnach ist ein Produkt ein Zusammenschluss von Modulen, Komponenten und Bauteilen, welche die Produktfunktion erfüllen. Dabei können, analog zur Definition von Elementen, materielle und immaterielle Produktanteile existieren (VDI, 1993).

Relation – Elemente können mittels Verbindungen, auch Relationen genannt, an ihren Schnittstellen verknüpft werden. Durch Relationen werden Energie-, Stoff- oder Datenflüsse beschrieben, die gerichteter und ungerichteter Art sein können (PAHL UND BEITZ, 2007).

System – Unter einem System wird der Zusammenschluss von Bauteilen und Komponenten sowie deren Relationen verstanden, die zur Erfüllung der Systemfunktion dienen, welche nur durch die zu Grunde liegende Struktur erfüllt werden kann. Durch In- und Output stehen Systeme mit ihrer Umwelt in Beziehung (LINDEMANN, 2009), wobei die Systemgrenze das betrachtete System von der Umwelt trennt, in die es eingebettet ist (NEGELE, 2006, S.66). Im Sprachgebrauch sind die Grenzen zwischen *Produkt* und *System* fließend (BUUR, 1990, S.19).

Flugzeugsystem – Unter Flugzeugsystemen werden im Kontext dieser Arbeit vorrangig mechatronische Systeme verstanden, die einen mechanischen, elektronischen und informationstechnischen Anteil haben und eine räumliche und funktionale Einheit bil-

den. Innerhalb dieser Systeme werden Funktionen mechanischer Komponenten durch elektrische Komponenten ersetzt und durch den informationsverarbeitenden Anteil intelligente bzw. autonome Eigenschaften geschaffen sowie erweiterte Funktionalität von Systemen erzeugt (ISERMANN, 2008).

Domäne – Im Kontext dieser Arbeit wird unter einer Domäne eine Fachabteilung innerhalb des Systementwicklungsprozesses verstanden, welche eine spezifische, von anderen Aufgabenbereichen klar abgegrenzte Tätigkeit ausführt.

Prozess – Unter einem Prozess wird sowohl der Ablauf der Entwicklungstätigkeiten innerhalb einer Domäne verstanden, als auch die Gesamtheit aller Tätigkeiten innerhalb eines ganzheitlich betrachteten Systementwicklungsprozesses.

Modell – Ein Modell ist die vereinfachende, formale Beschreibung eines geeigneten Ausschnitts des Betrachtungsgegenstandes (BUNGARTZ ET AL., 2009), welches die Grundlage für analytische Betrachtungen bildet, deren Durchführung mittels direkter Operationen am Original nicht möglich oder zu aufwändig wäre (SCHILLER, 2009). Im Kontext dieser Arbeit soll ein Modell zur Abbildung eines Systems in einem domänenübergreifenden Kontext dienen.

Metamodell – JECKLE (2000) definiert ein Metamodell als „ein Modell, das ein Modell beschreibt“. Der Begriff Metamodell ist aus dem griechischen Präfix *Meta* abgeleitet, welches im Zusammenhang dieser Arbeit mit „über“ übersetzt werden kann und stellt deshalb die formale Beschreibung eines Modells dar.

Typ – Ein Typ beschreibt die Eigenschaftsmerkmale von allen Elementen, die sich durch diese Eigenschaftsmerkmale hinreichend ausdrucksstark beschreiben lassen.

Attribut – Ein Attribut beschreibt ein Eigenschaftsmerkmal eines Typs.

Instanz – Eine Instanz ist die konkrete Ausprägung eines Typs. Eine Instanz wird durch die Belegung der Freiheitsgrade der Attribute eines Typs erzeugt (man spricht von Instanziierung).

Klasse – Analog zum Typ beschreibt eine Klasse die Eigenschaftsmerkmale gleichartiger Objekte. Die Begriffe Klassifizierung und Typisierung beschreiben jeweils den Vorgang des Einteilens der Objekte in die Klassen bzw. die Zuordnung zu den Typen (BANDOW UND HOLZMÜLLER, 2010).

Objekt – Ein Objekt stellt die Instanz einer Klasse dar und verhält sich entsprechend der Klassendefinition.

INHALT

1 EINLEITUNG

Die Entwicklung von technischen Produkten unterliegt über alle Wirtschaftsbereiche hinweg einem kontinuierlichen Wandel. Aufgrund von steigenden Kundenanforderungen an Funktion und Leistung bei gleichzeitig hohen Qualitätserwartungen ist eine deutlich anwachsende Komplexität von technischen Produkten zu verzeichnen. Um dieser Komplexität zu begegnen und am Markt erfolgreich zu bestehen, müssen Hersteller die Entwicklungskosten reduzieren und parallel die Entwicklungszyklen verkürzen. Die so entstehenden, vielschichtigen und vernetzten Entwicklungsabläufe stellen neue Herausforderungen an die eingesetzte Kommunikations- und Informationstechnologie. Seit Mitte der 90er Jahre haben sich die konstruktiven Entwicklungstätigkeiten der Ingenieure von der physikalischen zur digitalen Modellerstellung verschoben. Digitale Modelle erlauben eine deutlich kostengünstigere und schnellere Erstellung, Bearbeitung und Simulation von komplexen technischen Produkten und werden demnach intensiv genutzt.

Als Folge steht die Luftfahrtindustrie wie andere Hochtechnologiebranchen vor den Paradigmen der Komplexitätsbeherrschung, die zusätzlich durch die sprunghafte Entwicklung im Bereich eingebetteter Systeme und deren Verwendung im Flugzeugentwurf genährt werden. Eingebettete Systeme finden sich in der Luftfahrt in mechatronischen Systemen, die aus Komponenten mit mechanischen Bauelementen, Sensoren zur Erfassung des Systemzustandes, Aktuatoren zur Steuerung sowie einem Prozessor zur Informationsverarbeitung bestehen. Mechatronische Bauelemente beschränken sich dabei nicht auf die Flugzeugavionik, sie finden ebenso weitreichenden Einsatz in der Flugzeugkabine. Dabei ist festzustellen, dass sich eingebettete Systeme durch ihren hohen Integrationsgrad auszeichnen und somit vielschichtige Verknüpfungen zu anderen Systemen aufweisen.

Aufgrund der beschriebenen Systemkomplexität ist eine Vielzahl an Softwarelösungen im Einsatz, welche die fachspezifischen Entwicklungstätigkeiten optimal unterstützen. Da Entwicklungszyklen verkürzt werden sollen, müssen die beteiligten Prozesse paral-

lel ablaufen und stärker integriert werden, was durch die heterogene Infrastruktur aus isolierten Softwarewerkzeugen erschwert wird. Durch proprietäre Datenformate und begrenzte Exportfunktionalitäten der Software entstehen erhebliche Leistungsverluste in der Kommunikation und Zusammenarbeit von Abteilungen. Um eine signifikante Steigerung von Effizienz und Effektivität heutiger Entwicklungsprozesse zu erreichen, steht die Industrie vor der Herausforderung fachspezifische Einzeltätigkeiten ganzheitlich und methodisch zu integrieren.

1.1 Systementwurf im Wandel

Mit den neuen Herausforderungen des Systementwurfs beschäftigen sich die klassischen Forschungsrichtungen der Produktentwicklung im Maschinenbau und in zunehmenden Maße Disziplinen des Software- und Systems Engineering sowie integrierenden Disziplinen wie die Mechatronik. Allein diese Feststellung beschreibt die Vielschichtigkeit der Probleme, die sich im Rahmen einer hochintegrierten Entwicklung technischer Systeme ergibt. Die unzureichende Integration von Entwicklungsprozessen führt zu verschiedenen Schwierigkeiten: Zunächst sind die Aufgaben und Aktivitäten der Produktentwicklung und das Zusammenspiel von verschiedenen Entwicklungsdomänen nur zu geringen Teilen dokumentiert und koordiniert. Weiterhin werden durch domänenspezifische Werkzeuge Modelle erstellt, die relevante Aspekte eines technischen Systems unter dem domänenspezifischen Blickwinkel wiedergeben. Betrachtet man jedoch die einzelnen Aktivitäten, Werkzeuge und verwendeten Modelle aus einer bereichsübergreifenden Sicht, so besteht eine Vielzahl von wechselseitigen Verknüpfungen, die ebenfalls nur unzureichend abgebildet sind.

Vorhandene Lösungsansätze zur angesprochenen Problematik fokussieren unter den Schlagwörtern *Prozessintegration*, *Produktintegration* oder *Werkzeugintegration* jeweils nur Teilaspekte und stellen damit allenfalls Bausteine zur ganzheitlichen Lösung der integrierten Entwicklung technischer Systeme dar. Aufgrund der starken Fokussierung der heute etablierten Prozesse auf die verwendete Software und die hiermit erstellten Modelle, ist besonders die Integration von Teilmodellen aus unterschiedlichen Phasen des Entwicklungsprozesses eine große Herausforderung. Dies liegt an der meist hohen Modellkomplexität und den unterschiedlichen Sichten der Prozessbeteiligten auf ein Modell.

Die vorliegende Arbeit entwickelt in den nachfolgenden Kapiteln eine Methodik zur modellbasierten Entwicklung von Flugzeugsystemen, die zur effizienten Handhabung der genannten Herausforderungen der Systementwicklung beiträgt. Zunächst erfolgt

eine Analyse der aktuellen Situation des Systementwicklungsprozesses in der Luftfahrt sowie von Lösungsansätzen, die zur genannten Problematik beitragen. Aus den Ergebnissen wird ein Modell für die Beschreibung von Flugzeugsystemen generiert, wobei die Anforderungen einer praxisnahen Implementierung im Vordergrund stehen. Abschließend wird an einem industrienahen Anwendungsbeispiel die Umsetzbarkeit der modellbasierten Entwicklungsmethodik gezeigt.

1.2 Motivation und Zielsetzung

Flugzeughersteller stehen vor der Herausforderung der zunehmenden Systemkomplexität und Integrationstiefe ihrer Produkte. KRAUSE (2007) führt unter anderen Gründen für diese Entwicklung die disziplinübergreifende Systementwicklung, die Globalisierung, die Marktsituation sowie den Variantenreichtum an. Die Bedeutung von gesteigerter Qualität bei gleichzeitiger Halbierung der Entwicklungszeit wurde vom Rat für Luft- und Raumfahrt in Europa (engl.: Advisory Council for Aeronautics Research in Europe (ACARE)) in dessen strategischer Forschungsagenda (engl.: Strategic Research Agenda (SRA)) bereits 2004 beschrieben und als eine der großen Herausforderungen der europäischen Luftfahrtindustrie identifiziert. Um die Komplexität dieser Herausforderung beherrschbar zu machen, schlägt der Rat unter anderem vor, neue Produktentwicklungsmethoden anzuwenden, die flexible Produkte mit offenen Architekturen fördern. Als weitere strategische Maßnahme wurde vorgeschlagen, vermehrt Modellbildung und Simulation einzusetzen und kontinuierliche Verifikation und Validierung durch wissensbasierte Methoden zu implementieren (ACARE, 2004). Konkrete Beispiele für den zunehmenden Einsatz von komplexen, hochintegrierten Systemen in der Luftfahrt sollen die beschriebenen Herausforderung deutlicher machen und die Sinnhaftigkeit von modellbasierten Entwicklungsverfahren unterstreichen:

- Beispielhaft für die gestiegene Produktkomplexität in der Luftfahrtindustrie ist die zunehmende Elektrifizierung von Flugzeugsystemen zu nennen. Der Airbus A380 oder die Boeing 787 sind als Vertreter der More-Electric-Aircraft (MEA) mittlerweile am Markt vertreten, All-Electric-Aircraft-Architekturen (AEA) befinden sich in der Entwicklung. Pneumatische und hydraulische Systeme werden in diesen Konzepten durch elektrische Systeme ersetzt und es ergeben sich somit neue Anforderungen an deren Regelung und Energieversorgung. Die Vernetzung zwischen den Einzelsystemen nimmt zu und muss bereits in frühen Stadien des Entwurfs berücksichtigt werden. Das Interesse an elektrischen Flugzeugarchitekturen spiegelt sich in aktuellen Forschungsprogrammen wieder

(vgl.: Power Optimized Aircraft (FALEIRO, 2006); More Open Electrical Technologies (MOET, 2009); Systems for Green Operations (SGO-ITD, 2011)).

- Ebenso werden aufgrund zunehmender Rohstoffknappheit alternative Energieversorgungskonzepte für den Einsatz in Verkehrsflugzeugen untersucht (vgl. u.a. MOHR, 2006; PRATS UND LAVOIE, 2011). Die Integration von Konzepten wie verteilte Brennstoffzellen muss allerdings aufgrund ihrer sicherheitskritischen Bedeutung im Kontext des gesamten Flugzeuges ganzheitlich untersucht werden. Für detaillierte Analysen einer so tiefgreifenden Änderung der Systemarchitektur und der damit verbundenen Vielzahl an betroffenen Flugzeugsystemen, müssen Simulationen auf Flugzeugebene durchgeführt werden. Da die meisten Modelle von Flugzeugsystemen in domänenspezifischen Werkzeugen modelliert werden, die keine Schnittstellen zu anderen Softwareanwendungen besitzen, stellt sich die Analyse oder gar Optimierung von alternativen Energiekonzepten bislang als schwierig dar und ist Gegenstand aktueller Forschungsaktivitäten (u.a. SGO-ITD, 2011).
- Ein weiteres Beispiel für die zunehmende Integrationstiefe zukünftiger Flugzeugsysteme ist das Konzept aktiver Lastkontrolle (engl.: Active Control). Hierzu werden Systeme benötigt, welche zusätzliche Überwachungs- und Regelungsarchitekturen integrieren, um Strukturschwingungen zu dämpfen und damit Strukturbelastungen zu senken. Die Reduktion der aerodynamischen Lasten wird durch Steuerflächen und deren Aktuatoren umgesetzt. Ein solch hochintegriertes Konzept hat entsprechende Auswirkungen auf die Auslegung der aerodynamischen Steuerflächen, deren Regelungsmechanismen sowie die umgebende Struktur. Die Systemkomplexität ist aufgrund ihrer Erstreckung über mehrere Domänen äußerst hoch und muss im Vorentwurf entsprechend abgebildet werden (MOHR, PAULUS, BAIER UND HORNUNG, 2011), um robuste Produkte in angemessener Entwicklungszeit auf den Markt bringen zu können.

Die hier aufgeführten Beispiele aktueller Forschung in der Luftfahrt haben einen gemeinsamen Anknüpfungspunkt: ein komplexes Subsystem wird in ein komplexes Gesamtsystem integriert. Aufgrund der hohen Anzahl von Schnittstellen und Querbezügen innerhalb von Flugzeugsystemen ist die Bestimmung der Systemleistung eines Subsystems und dessen Auswirkungen im Gesamtsystem äußerst komplex. Dies liegt vor allem an der unzureichenden Möglichkeit, domänenübergreifende Information zentral zu speichern, der fehlenden integrativen Betrachtung der Werkzeugketten sowie der unzureichenden Koordination von Arbeitsabläufen. Um diesen Umständen und

der damit verbundenen Komplexität auf Produkt- und Prozessebene zu begegnen, propagiert diese Arbeit den Einsatz von domänenübergreifenden Metamodellen. Unter einem Metamodell wird im Rahmen dieser Arbeit die informationstechnische Abbildung eines Systems verstanden, welche domänenübergreifende Informationen durchgängig erfasst, in Relation setzt und unterschiedliche Sichten auf ein System ermöglicht. Diese Arbeit formuliert die Anforderungen an ein Metamodell aus vorhandenen Forschungsansätzen und aktuellen Prozessen der Systementwicklung in der Luftfahrt. Es wird ein Metamodell vorgestellt, das sich aus unabhängigen und domänenspezifischen Partialmodellen zusammensetzt und die Grundlage für eine Datenbank darstellt, die domänenübergreifende Informationen abbilden kann. Die Wirkungsweise des Metamodelleinsatzes wird durch die Integration in Arbeitsabläufe der Systementwicklung anhand eines Anwendungsbeispiels gezeigt.

1.3 Struktur der Arbeit

Diese Arbeit besteht aus vier Teilen mit insgesamt sieben Kapiteln. Der erste Teil – Einführung – beschreibt in Kapitel 1 den Hintergrund, die Motivation und die Zielsetzung sowie die Struktur der Arbeit. In Kapitel 2 werden aktuelle Entwicklungsabläufe von Flugzeugen und Flugzeugsystemen sowie bestehende Defizite diskutiert.

Im zweiten Teil – Grundlagen – werden zunächst in Kapitel 3 aktuelle Forschungsansätze, Methoden und Rahmenbedingungen der Systementwicklung vorgestellt, um daraus Potenziale und Handlungsempfehlungen abzuleiten. Als Folge der identifizierten Anforderungen an die zukünftige Systementwicklung wird in Kapitel 4 der Entwurf eines Metamodells zur Beschreibung von Flugzeugsystemen beschrieben.

Der dritte Teil dieser Arbeit – Analyse – zeigt in Kapitel 5 wie das formale Konstrukt des Metamodells in einer Entwurfsumgebung implementiert werden kann. Dazu werden die benötigte Infrastruktur sowie die Schnittstellengestaltung von vorhandenen Entwicklungswerkzeugen zur Metamodellebene beschrieben. Nach ROPOHL (1970) hat angewandte Forschung „stets ein praktisches und ein theoretisches Ziel, wobei ersteres nur über letzteres erreichbar scheint. Da theoretische Erkenntnis bei angewandter Forschung nicht um ihrer selbst angestrebt wird, müssen sich die theoretischen Erkenntnisziele aus praktischen Gestaltungszielen ableiten lassen“. Um diesen Kreis zu schließen, zeigt Kapitel 6 einen konkreten Anwendungsfall der Methode am Beispiel der Entwicklung einer Flugzeugklimaanlage.

Im letzten Teil – Zusammenfassung – werden in Kapitel 7 die Ergebnisse der Arbeit diskutiert und ein Ausblick für nachfolgende Arbeiten gegeben. Eine Übersicht der Kapitelstruktur ist nachfolgend in Abbildung 1-1 dargestellt.

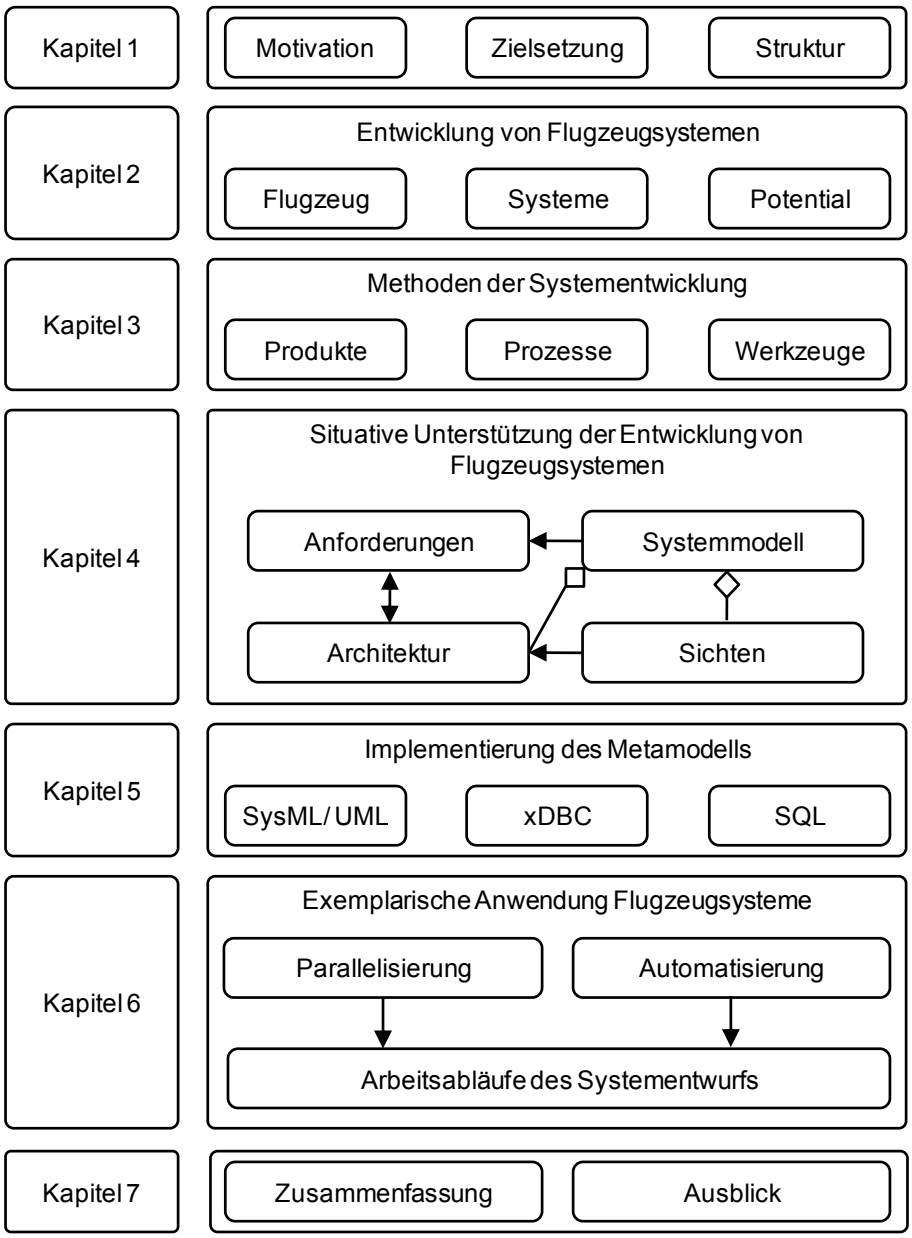


Abbildung 1-1 Struktur der Arbeit

2 ENTWICKLUNG VON FLUGZEUGSYSTEMEN

Im nachfolgenden Kapitel wird zunächst der Flugzeugbau im Spannungsfeld geänderter Umweltbedingungen diskutiert, bevor der Stand der Technik der in der Luftfahrtindustrie eingesetzten Methoden und Entwicklungsansätze untersucht wird. Dabei steht vorerst der Flugzeugentwurfsprozess im Vordergrund, später wird auf die Systementwicklung im Speziellen eingegangen. Dieses Kapitel soll ein grundsätzliches Verständnis über die Entwurfskomplexität von Luftfahrzeugen und deren Systemen vermitteln und das Potenzial modellbasierter Systementwicklung unterstreichen.

2.1 Flugzeugbau im Spannungsfeld geänderter Umweltfaktoren

Der formale Vorgang des Entwickelns hat sich in den letzten Jahrzehnten nur evolutionär verändert und es stellt sich die Frage, warum neue Ansätze in der Entwicklung von Flugzeugsystemen notwendig werden. Diese Frage lässt sich mit den geänderten Bedingungen erklären, in denen sich die globale Wirtschaft und damit die Flugzeugindustrie befinden:

Verkürzte Entwicklungs- und Einführungszeit: Durch Globalisierungseffekte wie der Zunahme von Produktanbietern oder verbesserter Informationsverfügbarkeit über am Markt vorhandene Produkte, sind Endverbraucher sehr gut über Produkte und deren Alternativen informiert. Daraus entsteht ein Leistungsdruck auf Hersteller, stets technisch aktuelle und kostengünstige Produkte im Vergleich zur Konkurrenz am Markt anzubieten (SHEA, 2011). Vor allem in der Luftfahrtindustrie ist dieser Zeitdruck besonders groß, da hier eine Reaktion auf Marktschwankungen aufgrund der Komplexität des Produktes sowie restriktiver Zulassungsverfahren besonders schwierig ist. Es besteht demnach die konkrete Forderung nach Werkzeugen, die eine gesteigerte Flexibilität und eine Verkürzung von Entwicklungszeiten ermöglichen. Ein Beispiel aus der nahen Vergangenheit ist die Dassault Falcon X. Durch den gezielten Einsatz von Product Lifecycle Management (PLM) Werkzeugen in diesem Flugzeugent-

wurfsprogramm konnte die Dauer der Erstmontage um 50% reduziert werden (MURPHY, 2005).

Kostenreduktion: Die Zunahme von Funktionalität und Qualität bei gleichbleibenden oder sogar sinkenden Kosten entspricht der heutigen Erwartungshaltung des Käufers eines technischen Produktes. Ein Grund für dieses Paradigma ist die gesteigerte Effizienz von Prozessen und die damit verbundenen Kostenreduktion bei den Herstellern (SHEA, 2011). Laut EHRLENSPIEL ET AL. (2007) verursacht die Entwicklung und Konstruktion technischer Produkte einen vergleichsweise geringen Anteil der Produktkosten (ca. 9%). Allerdings determiniert diese Produktentwicklungsphase ca. 70% der Produktkosten und stellt demnach den wichtigsten Hebel zur Kostenreduktion der Produktentwicklung dar. Um diesen Hebel möglichst effizient zu nutzen, werden vermehrt computergestützte Modelle in der Entwicklung eingesetzt, welche Simulation und Optimierung von Systemen ermöglichen (DOLEZAL, 2008; SHEA, 2011). Ein weiterer Vorteil von Modellen ist die Konsistenz und Redundanzfreiheit von Produktdaten (IRLINGER, 1999, S.66), welche das Systemverständnis verbessern, Fehlerfreiheit ermöglichen und somit zur Kostenreduktion beitragen.

Qualitätssteigerung: Steigende Qualität äußert sich unter anderem durch gesteigerte Funktionalität oder Variabilität, welche beispielhaft durch den Einsatz von eingebetteten Systemen oder innovativer Produktgestaltung ermöglicht werden. Beide Ansätze basieren auf zunehmender Nutzung von computergestützten Entwicklungswerkzeugen sowie der tieferen Integration von Entwicklungsdomänen und dem gezielten Einsatz von wissensbasiertem Vorgehen (SHEA, 2011). Ein oft zitiertes Beispiel für die computergestützte Flugzeugentwicklung ist die Entstehung der Boeing 777. Folgende Ziele im Hinblick auf die Produktqualität konnten durch den gezielten Einsatz des CAD/PLM Werkzeuges CATIA V5 erzielt werden (SABBAGH, 1996):

- Kein physikalischer Prototypenbau
- Beseitigung von über 3000 Montageinterferenzen
- Verbesserter Zusammenbau des Rumpfes durch minimierte Bauteiltoleranzen
- 90% Reduktion von Änderungsaufträgen

Zunehmende Komplexität von Systemen: Die Zunahme der Systemkomplexität ist durch den Markt, die Entwicklungsprozesse, die Entwicklungsorganisation sowie die Systemstruktur bedingt (LINDEMANN, 2009). Zum einen werden immer mehr Varianten eines Produktes hergestellt um einen breiteren Markt zu bedienen, zum anderen steigt die Komplexität der einzelnen Varianten. Um eine große externe Variantenvielfalt bei möglichst geringer, handhabbarer Vielfalt darstellen zu können, werden

Methoden wie Modular Function Deployment (ERICCSON UND ERIXON, 1999) oder Plattformdesign (beschrieben in SIMPSON, 2004) eingesetzt. Methoden zu Beherrschung von Produktkomplexität (LINDEMANN, 2009) und der damit einhergehenden Prozesskomplexität (WILDEMANN, 2004) finden vermehrt Anwendung in den Unternehmen. Allen Methoden gemeinsam ist die notwendige Daten- und Wissensbasis. Je komplexer und variantenreicher sich das Produkt gestaltet, desto größer wird der Bedarf die entsprechenden Daten sinnhaft zu nutzen und zu verwalten. Vor diesem Problem stehen auch heute noch viele Unternehmen trotz großer Fortschritte im Bereich PLM (RANGAN ET AL., 2005).

Verteilte Entwicklung und Produktion: Um konkurrenzfähige Produkte am Markt anbieten zu können, nutzen viele Unternehmen Vorteile wie geringere Produktionskosten an einem Standort und wissenschaftliche Exzellenz an einem anderen Standort. Die Standorte von global agierenden Unternehmen werden bewusst so gewählt, dass eine maximale Effizienz der Prozesse ermöglicht wird (z.B.: 24/7 Entwicklung). Hierzu sind Werkzeuge notwendig, welche die Kommunikation, Koordination und Kooperation zwischen diesen verteilten Teams ermöglichen (SHEA, 2011).

Elektrifizierung der Flugzeugsystemarchitektur: Aufgrund der zunehmenden Rohstoffknappheit fossiler Brennstoffe finden in Forschung und Entwicklung große Anstrengungen statt, effizientere Flugzeuge zu entwerfen. Im Bereich der Flugzeugsysteme bedeutet dies vor allem eine zunehmende Elektrifizierung der Systeme und damit dem gesteigerten Einsatz mechatronischer Systeme. Dieser Paradigmenwechsel innerhalb der Systemarchitektur resultiert aus der zu erwartenden Gesamtenergieeffizienz (DEROUINEAU, 2009) und dem verbesserten Leistungsgewicht von elektrischen Systemen im Vergleich zu herkömmlichen hydraulischen oder pneumatischen Systemen. Auch MOIR UND SEABRIDGE (2003) treffen die Aussage, dass die Mehrzahl der heute entwickelten Flugzeugsysteme einen elektronischen Anteil besitzt (Kontrollfunktion) und dieser wiederum entsprechende Softwareentwicklung bedingt. JÄNKER (2007) und DEROUINEAU (2009) bestätigen den Trend zu More- bzw. All-Electric-Aircraft-Architekturen im Kontext mechatronischer Systeme, welcher vor allem durch die Entwicklungsrate von Computerressourcen und Leistungselektronik ermöglicht wird. Der Einsatzbereich von mechatronischen Komponenten im Flugzeugbau erstreckt sich dabei nahezu über alle Flugzeugsysteme wie Klimaanlage, Enteisungsanlage, Flugsteuerung, Fahrwerk oder Kabinensysteme (Abbildung 2-1).

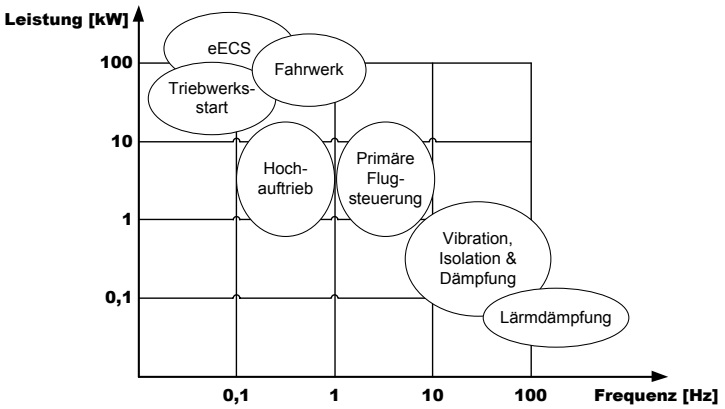


Abbildung 2-1 Einsatzbereiche mechatronischer Flugzeugsysteme
Quelle: JÄNKER (2007), leicht modifiziert

Entwicklungskomplexität mechatronischer Produkte: Durch die veränderte Funktionsaufteilung der Systeme – von der mechanischen Funktion hin zur elektronisch realisierten Funktion – ergeben sich neue Anforderungen an die Systementwicklung. Die herkömmliche Unternehmensstruktur von luftfahrttechnischen Unternehmen orientiert sich an der Struktur der ATA-Kapitel, einer Einteilung von Flugzeugsystemen, die ausschnittsweise in Tabelle 2-1 wiedergegeben werden.

Tabelle 2-1 Übersicht der ATA- Kapitel
(Auszug aus FAA, 2011)

Kapitel	Bezeichnung
ATA 21	Environmental Control System (ECS)
ATA 24	Electrical Power System (EPS)
ATA 27	Flight Control Systems
	Primary Flight Control Systems (PFCS)
	Secondary Flight Control Systems bzw. High Lift Systems (HLS)
ATA 28	Fuel Systems (FUS)
ATA 29	Hydraulic Power Systems (HPS)
ATA 30	Ice Rain Protection Systems
	Wing Ice Protection System (WIPS)
	Nacelle Anti Ice (NAI)
ATA 32	Landing Gear Systems (LGS)
ATA 36	Bleed Air Systems bzw. Pneumatic Power Systems (PPS)
ATA 49	Auxiliary Power Unit (APU)

Durch die beschriebene Änderung der Systemarchitektur werden vormals getrennte Systeme miteinander verknüpft oder werden einem anderen ATA-Kapitel zugeordnet. Die Entwicklungsprozesse müssen an diese neuen Herausforderungen adaptiert und stärker integriert werden. Unter Prozessintegration ist beispielsweise die Synchronisierung mit dem Flugzeugprogramm zu verstehen, welche unter anderem zum Einsatz sogenannter Systemarchitekten (auch: Systemingenieur, Systemintegrator) führt. Diese Personen sollen den globalen und funktionalen Überblick über Systeme und deren Schnittstellen zum Gesamtsystem besitzen (SPUR UND KRAUSE, 1997; WEILKIENS,

2008). Doch nicht nur die Integration von Prozessen nimmt zu, auch Bauteile mechatronischer Produkte sind stärker vernetzt und auch hier entstehen neue Anforderungen im Bereich der Systemintegration. Insbesondere die Elektromagnetische Verträglichkeit (EMV), die Bereitstellung von Datenlinks, die Stromversorgung und die strukturelle Anbindung der Systeme sind zu erwähnen. Mit der zunehmenden Anzahl von Sensoren, Aktuatoren, Schaltern oder Regelungen wächst die Zahl der erforderlichen Kabelverbindungen stark an, was zu höheren Kosten, Gewicht, Kontakten und Bauraumanforderungen führt (ISERMANN, 2008, S.20). Der Einsatz von neuen digitalen Bus-Systemen reduziert zwar diese Nachteile, die Komplexität der Verkabelung von Flugzeugsystemen bleibt dennoch ein aktuelles Forschungsthema (FIGLAR UND MOHR, 2011). Schließlich wird die Zuverlässigkeitsanalyse zum wichtigen Bestandteil der Systementwicklung aufgrund der größeren Anzahl an Komponenten, die im Vergleich zu rein mechanischen Systemen ein anderes, meist ungünstigeres Ausfallverhalten besitzen (ISERMANN, 2008, S.21).

Die Begrenzung auf mechatronische Systeme scheint zunächst den thematischen Rahmen dieser Arbeit einschränken zu wollen. Doch das Gegenteil ist der Fall: Die Entwicklung von mechatronischen Systemen ist vielmehr eine integrative Aufgabe von vormals getrennten Entwicklungsdisziplinen (ISERMANN, 2008, S.33). Als Folge stehen Systementwickler vor der Aufgabe, verschiedene Entwicklungsdomänen sowie deren Anforderungen und Fachwissen integrieren zu müssen. Als Folge des zunehmenden Einsatzes von mechatronischen Systemen und einer stärkeren Elektrifizierung müssen vorhandene Entwicklungsprozesse an die neuen Herausforderungen kontinuierlich angepasst werden.

2.2 Merkmale des Flugzeugentwurfs

Wie jedes technische Produkt durchläuft ein Flugzeug verschiedene Phasen während seiner Entwicklung, die sich von der Produktidee bis zum Markteintritt des Produktes erstrecken. Unter einem Entwicklungsprozess, kurz Prozess, werden im Rahmen dieser Arbeit die Gesamtheit aller Tätigkeiten und deren Abhängigkeiten verstanden, die zur Erstellung eines Produktes oder Systems notwendig sind. Zur besseren Übersichtlichkeit, werden spezifische Tätigkeiten einer entsprechenden Entwicklungsphase, kurz Phase, zugeordnet. Es existieren verschiedene Modelle für die Beschreibung von Flugzeugentwurfsprozessen, die sich hauptsächlich in der Wahl des Granularitätsniveaus und damit in der Bezeichnung und Anzahl der Phasen unterscheiden (vgl. MOIR UND SEABRIDGE, 2003; PARDESSUS, 2004; LIZARRALDE, 2007; VILSMEIER, 2011). Im Folgenden wird der Entwicklungsprozess für Flugzeuge in Anlehnung an LIZARRALDE

(2007) dargestellt, um einen Überblick über relevante Prozessabschnitte sowie deren Inhalte und Ergebnisse zu erhalten.

Machbarkeitsphase – Der Wunsch nach einem neuen Produkt – hier einem Flugzeug – resultiert zumeist aus einem Markbedürfnis (engl.: market pull) oder einem technologischen Fortschritt bzw. dem Verwendungsdruck einer Innovation (engl.: technology push) (MEIER, 2005). Auf eine konkrete Produktidee folgt eine umfassende Marktanalyse und Festlegung der Anforderungen an das neue Produkt. In dieser Phase werden die Machbarkeit und die wichtigsten Parameter festgelegt (TORENBEEK, 1982). Dieser Schritt ist von großer Bedeutung, da die hier definierten Anforderungen die Produktfunktionen festlegen, welche wiederum die Produktstruktur bedingen. In der Folge werden verschiedene Varianten einer Produktstruktur untersucht, entsprechende Wirtschaftlichkeitsrechnungen sowie Vergleiche mit Konkurrenzprodukten durchgeführt. Es werden potenzielle Kunden und Zulieferer zu alternativen Detailkonzepten befragt und die Technologiereife der gewählten Systeme abgesichert. Den abschließenden Meilenstein dieser Phase stellt die Auswahl und Definition des Basiskonzeptes dar.

Konzeptphase – In der Konzeptphase wird das Basiskonzept auf Flugzeugebene kontinuierlich weiter detailliert. Dazu gehören vielfältige Analysen technischer Aspekte (u.a. Aerodynamik, Strukturbelastung) sowie operationeller und wirtschaftlicher Aspekte (u.a. Beladezeiten, Betriebskosten). In der Konzeptphase arbeiten verhältnismäßig wenige Fachleute verschiedener Fachdisziplinen zusammen und die Kommunikation erfolgt noch zu einem großen Teil auf sprachlicher Ebene. Die Konzeptphase wird mit der Validierung des Detailkonzeptes abgeschlossen (Meilenstein M5).

Definitionsphase – In dieser Phase wird das Detailkonzept des Flugzeuges um Analysen auf System- und Komponentenebene erweitert. Dazu gehören die Simulationen und Optimierung von Systemen und Subsystemen sowie Integrationsstudien. Weiterhin wird festgelegt, welche Bauteile und Systeme eigenverantwortlich hergestellt werden und welche Bauteile in Fremdvergabeumfänge fallen. Alle Tätigkeiten werden Prozessen zugeordnet, welche die Termineinhaltung garantieren sollen. Die Synchronisierung zwischen Prozesspartnern erfolgt über einen definierten Änderungsprozess. Zusätzlich werden alle benötigten Betriebsmittel zur Produktentwicklung und Produktion definiert, hergestellt oder beschafft. Schließlich werden Vorserienmodelle gefertigt und verschiedene Versionen dieser Prototypen getestet. Ziel ist es, die Leistung und Zuverlässigkeit der Produkte zu testen, um eventuelle Änderungen des Endproduktes implementieren zu können. Die Art der verwendeten Prototypen im Flugzeugbau wandelt sich von physikalischen zu digitalen Prototypen (DOLEZAL, 2008). Am

Beispiel der Dassault Falcon X konnte gezeigt werden, dass die Entwicklung und Zulassung ohne die Verwendung von physikalischen Prototypen auskommen kann (MURPHY, 2005). Die damit verbundenen Zeit- und Kostenvorteile sind signifikant und unterstreichen das Potenzial digitaler Modelle in der Herstellung komplexer technischer Produkte. Während der Definitionsphase arbeitet eine Vielzahl von Fachleuten unterschiedlicher Spezialisierung zusammen und der Informationsaustausch ist zunehmend digitaler Art. Der abschließende Meilenstein dieser Phase ist die Spezifikation aller Flugzeugsysteme und damit die vollständige Definition des Fluggerätes.

Entwicklungsphase – Während der Entwicklungsphase beginnen die Komponentenfertigung und der erste Zusammenbau von Systemen. Es entstehen die ersten physikalischen Engineering Mockups und folglich können die ersten realen Systemtests erfolgen. Weiterhin dient der Produktionshochlauf zur Justierung der Abläufe der Serienfertigung und dem Training der Produktionsbelegschaft. Während dieser Phase werden Produkte bereits von bestimmten Kunden getestet und die letzten Diskrepanzen zwischen Produktfunktion und Kundenanforderung beseitigt. Der Übergang zur Serienfertigung ist meist fließend und markiert den Zeitpunkt, ab dem das Endprodukt am Markt verfügbar wird (SPUR UND KRAUSE, 1997).

Die eben beschriebenen Phasen können klar durch Meilensteine (auch: Quality Gates) voneinander getrennt werden und stellen hierbei Entscheidungspunkte sowie spezifische Zieltermine dar. Um Prozesse handhabbar zu machen, werden verschiedene Hierarchieebenen eingeführt. Dementsprechend sind Meilensteine verschiedener Hierarchie vorhanden, wobei ein Meilenstein höherer Hierarchie nur erreicht werden kann, wenn alle zugehörigen Meilensteine niedrigerer Hierarchie erreicht worden sind. Weiterhin sind zur besseren Unterscheidung verschiedene Klassen von Meilensteinen vorhanden. Dazu gehören vor allem technische und kommerzielle Meilensteine. Die Zielvorgaben der Meilensteine dienen weiterhin der Synchronisierung von Abteilungen oder von Hersteller und Zulieferer in parallelen Entwicklungsprozessen.

Abbildung 2-2 zeigt exemplarisch den „Design-New-Aircraft“-Prozess von Airbus mit den Hauptprozessen und den technischen Meilensteinen, welche die Prozesse miteinander verbinden.

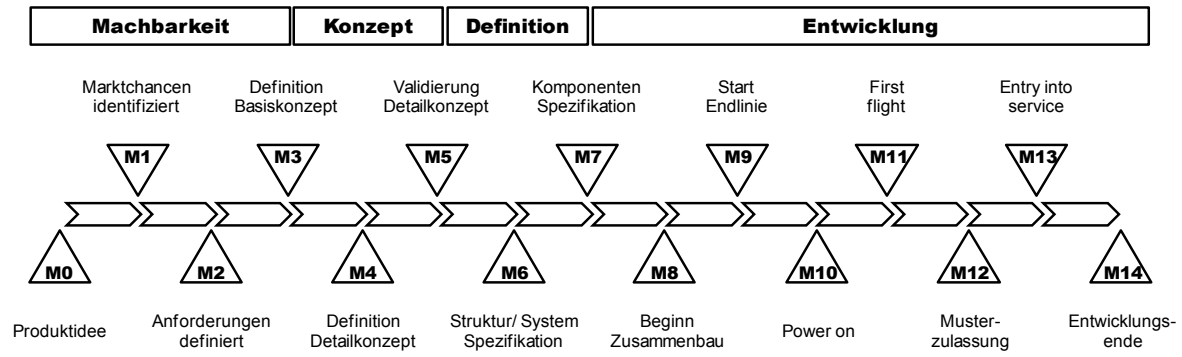


Abbildung 2-2 Design-New-Aircraft-Prozess von Airbus
Quelle: LIZARRALDE (2007), SCHMIDT (2009)

2.3 Merkmale des Entwurfs von Flugzeugsystemen

Der Systementwurf stellt einen Vorgang dar, der in den übergeordneten Flugzeugentwurfsprozess integriert und mit diesem synchronisiert werden muss. Die Systementwicklung wird demnach durch die Abläufe der kundenindividuellen Flugzeugmusterfertigung beeinflusst und muss die geforderte Qualität von Prozessen und Ergebnissen innerhalb dieser Grenzen gewährleisten. Der (Sub-)Systementwurf obliegt deshalb stärkeren Restriktionen und findet unter größerem Zeitdruck statt, als dies bei Baugruppen auf höherem Abstraktionsniveau der Fall ist. Weiterhin ist festzustellen, dass der generische Ablauf der Flugzeugentwicklung (Kapitel 2.2) keine explizite Vorgabe für die iterativen Vorgänge zwischen verschiedenen Entwicklungsdomänen macht. Da gerade hier ein besonders großes Wertschöpfungspotenzial zu erreichen ist (BROY, 2010), soll im Folgenden auf die Spezifika der Systementwicklung genauer eingegangen werden.

2.3.1 Arbeitsabläufe

Für die Entwicklung eines Systems arbeiten verschiedene Fachabteilungen mit spezifischen Verantwortlichkeiten zusammen. Diese Domänen umfassen zunächst alle gestalterischen Entwicklungstätigkeiten mit einem konkreten physikalischen Endergebnis. Dazu kommen unterstützende Tätigkeiten, welche vor allem die Administration und Fertigung umfassen. Um diese Tätigkeiten zu koordinieren werden Ablaufdiagramme (engl.: flow chart, sequence chart, swim lane diagram) erstellt, welche die Einzeltätigkeiten der Domänen in einen zeitlichen Bezug zueinander setzen und die Zeitpunkte abbilden, zu denen eine Überprüfung der erreichten Ergebnisse stattzufinden hat (Meilensteine, alternativ: Quality Gates (QG)). Eine beispielhafte Darstellung eines Ablaufdiagramms findet sich in Abbildung 2-3.

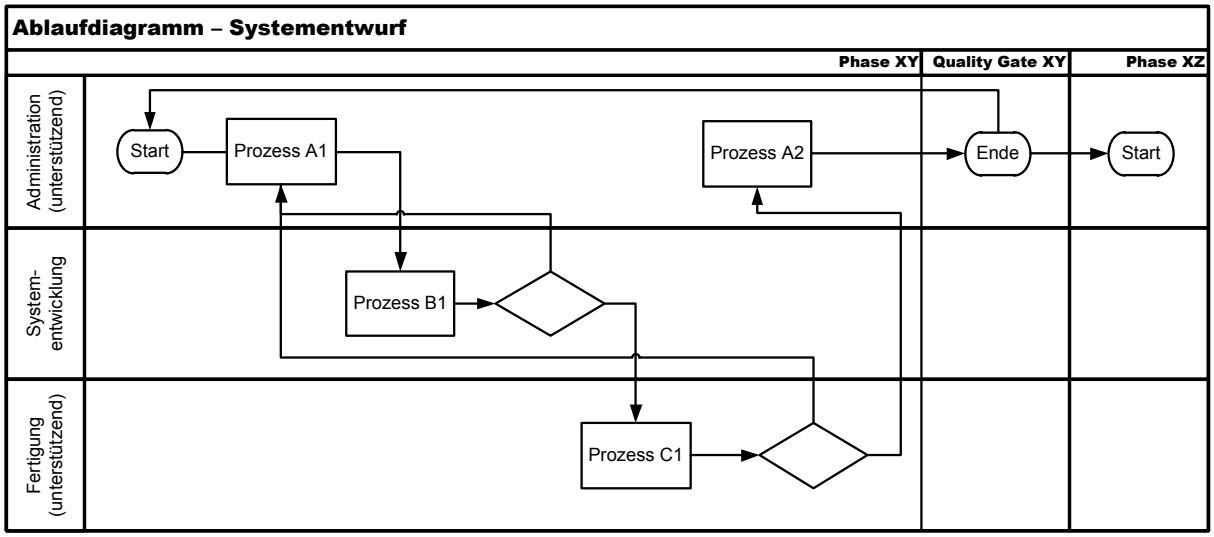


Abbildung 2-3 Struktur eines Ablaufdiagramms

Es ist festzustellen, dass sich durch Ablaufdiagramme lediglich die sequentielle Abfolge von Prozessen darstellen lässt, wobei Folgeprozesse von Informationsverfügbarkeit oder Entscheidungen abhängig sind. Die Darstellung von parallelen Entwicklungstätigkeiten ist nur eingeschränkt möglich. Bevor Möglichkeiten zur Parallelisierung von Prozessen und deren Darstellungsmöglichkeiten diskutiert werden, sollen die Prozesse identifiziert werden, die für den Kontext dieser Arbeit eine besondere Relevanz aufweisen. Dazu sollen die Gesichtspunkte einer möglichen Klassifizierung von Prozessen diskutiert werden. KÜHLBERGER (2002) schlägt die Kriterien *Strukturiertheit, Ausführungshäufigkeit, Komplexität, Dauer, Veränderlichkeit, Arbeitsteiligkeit* und *Automatisierbarkeit* zur Klassifizierung von Prozessen vor.

Die *Strukturiertheit* beschreibt das Maß, in dem ein Prozess in klar abgrenzte Tätigkeiten mit fester Abfolge unterteilt ist. Unstrukturierte Prozesse hingegen zeichnen sich durch ein hohes Maß an spontanen Tätigkeiten und Entscheidungen aus. Entwicklungsprozesse von technischen Produkten zählen zu den semistrukturierten Prozessen bei denen eine globale Ablaufstruktur existiert, jedoch einzelne Phasen enthält, deren Verlauf nicht planbar ist (GÖTZER, 1997). Gewisse Phasen der Produktentwicklung wie die (kreative) Synthese einer Produktidee in der Konzeptphase sind nur zu geringen Teilen strukturiert, während spätere Phasen der Entwicklung ein hohes Maß an Strukturiertheit aufweisen müssen. Die Flugzeugentwicklung weist zwar einen gut strukturierten Makrozyklus auf, die einzelnen Phasen laufen jedoch hochgradig dynamisch und iterativ ab (RAYMER, 2006).

Die *Ausführungshäufigkeit* beschreibt die Anzahl der Wiederholungen eines spezifischen Prozesses innerhalb einer Entwicklungsphase. Je größer die Ausführungshäufig-

keit, desto größer ist das Potenzial für den Einsatz von Rechnerunterstützung in der Entwicklung oder der Automatisierungstechnik in der Montage.

Ein wichtiges Merkmal eines Prozesses ist dessen *Komplexität*. Diese wird unter anderem durch die Anzahl der beteiligten Personen sowie den Vernetzungsgrad der Tätigkeiten beschrieben. Weitere Kriterien sind die benötigten Fähigkeiten der Personen und der Grad notwendiger Werkzeugunterstützung. Demnach besitzen die meisten Entwicklungsprozesse in der Luftfahrt ein sehr hohes Maß an Komplexität, da sie vermehrt von vielen Personen unterschiedlicher, fachlicher Qualifikation bearbeitet werden. Weiterhin bestehen ein sehr hoher Vernetzungsgrad zwischen Disziplinen und eine große Anzahl an domänenspezifischer Software. Die Inhomogenität der Soft- und Hardwarelandschaft führt zu einer Vielzahl an notwendigen Schnittstellen (SPUR UND KRAUSE, 1997), welche häufig durch Datenverlust, Einbüßen von Genauigkeit, hohen Wartungsaufwand und entsprechende Kosten gekennzeichnet sind (SHEA, 2011).

Die *Dauer* als weiteres Merkmal von Prozessen liegt bei der Entwicklung von technischen Produkten aufgrund der inhärenten Komplexität relativ hoch. Für Flugzeugprogramme finden sich Werte zwischen wenigen Jahren bis mehreren Jahrzehnten (MURPHY, 2005; HORNUNG, 2011). In Bezug auf die Dauer von Entwicklungsprozessen von Flugzeugsystemen besteht heute noch großes Potenzial durch einen erhöhten Grad an Parallelität und Automatisierung von Prozessabläufen. Grundlage hierfür ist die Schaffung stark strukturierter Prozesse und klar definierten Meilensteinvorgaben.

Weiterhin beschreibt die *Veränderlichkeit* eines Prozesses dessen Adaptionsbedarf an spezifische Entwicklungsaufgaben. Je weniger ein Prozess an spezifische Tätigkeiten angepasst werden muss, desto robuster ist er gegenüber Änderungen und kann demnach leichter in automatisierte Workflows integriert werden.

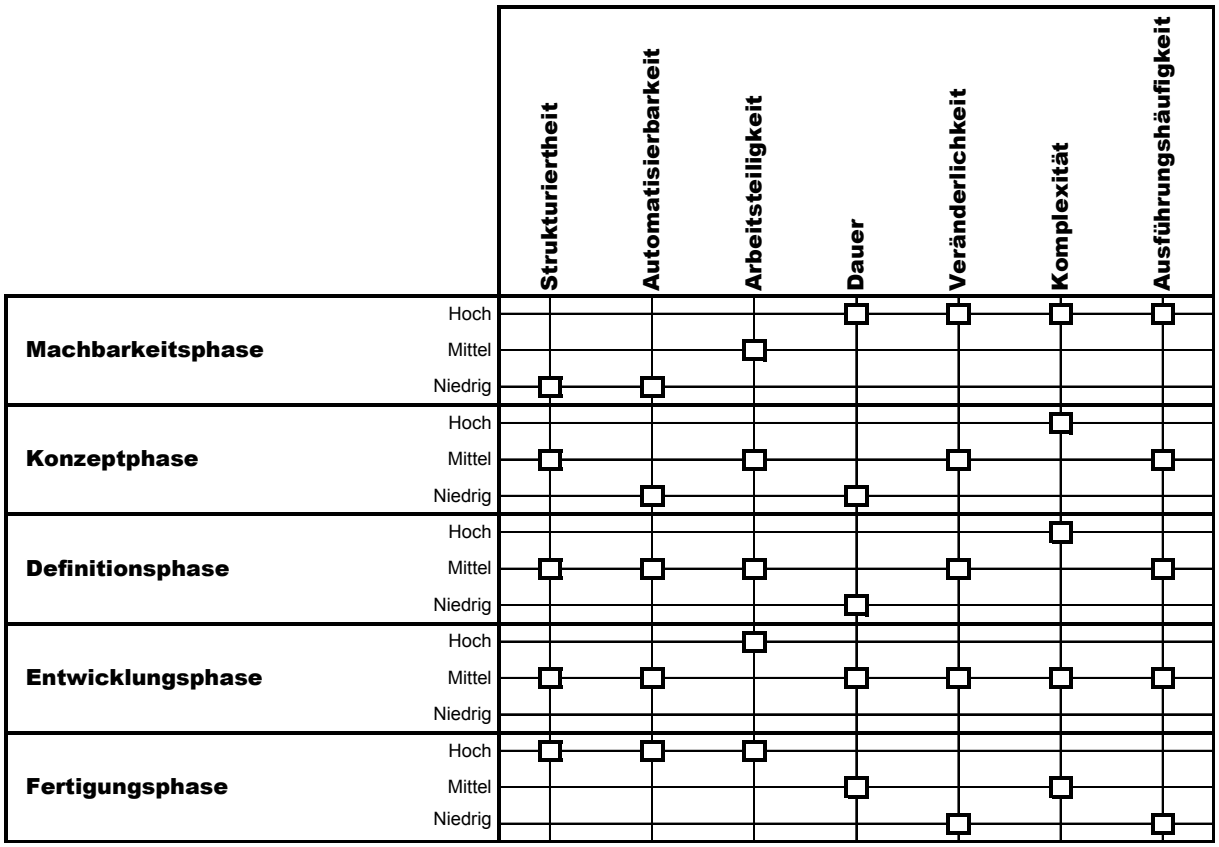
Arbeitsteiligkeit beschreibt, wie sich Prozesse im Ganzen oder in Teilen auf verschiedene Personen oder Abteilungen aufteilen. Gerade im Bereich der Entwicklung technischer Produkte wie mechatronische Flugzeugsysteme ist eine hohe Arbeitsteiligkeit zu verzeichnen. Ein Grund ist die hohe Systemkomplexität, welche den Einsatz einer Vielzahl von Experten bedingt und eine tiefe Integration von Mensch und Prozess hervorruft.

Das letzte Kriterium der Prozessklassifikation stellt die *Automatisierbarkeit* dar, welche die Möglichkeit einer kompletten oder teilweisen rechnergestützten Durchführung von Prozessen beschreibt.

Mit Hilfe der eben beschriebenen Charakteristika sollen die Prozesse des Flugzeugentwurfs (Kapitel 2.1) auf deren Eignung für den Einsatz von Systemmodellen über-

prüft werden. Dazu wurde eine qualitative Bewertung der einzelnen Prozesse vorgenommen (Tabelle 2-2). Es ist der hohe Komplexitätsgrad von der Konzeptphase bis zur Entwicklungsphase zu erkennen, welcher in der Vielzahl von Freiheitsgraden und dem hohen Vernetzungsgrad der Tätigkeiten begründet ist. Die Anzahl der Entwickler sowie deren Arbeitsteiligkeit ist in diesen Phasen im Vergleich zu späteren Produktlebensphasen relativ gering; der Einfluss von Entscheidungen jedoch sehr groß.

Tabelle 2-2 Charakteristika des Entwurfs von Flugzeugsystemen



Die geringe Strukturiertheit der Vorentwicklungsphasen bedingt die schwierige Automatisierbarkeit von Teilprozessen. Die relativ hohe Ausführungshäufigkeit und Dauer der Tätigkeiten dieser Phasen macht die domänenübergreifende Werkzeugunterstützung jedoch attraktiv. Erst spätere Phasen der Produktentwicklung wie die Produktionsphase weisen einen hohen Grad an Strukturiertheit, Automatisierbarkeit und Arbeitsteiligkeit auf. Die zugrundeliegenden Prozesse sind weniger komplex, veränderlich und werden weitaus weniger häufig ausgeführt. Sie greifen auf bereits definierte Produktdaten zurück, die nur marginalen Änderungen unterliegen. Diese Phasen des Lebenszyklus eines Flugzeugsystems werden heute bereits durch die vorhandene Integration von DMU und PDM-Systemen sehr gut unterstützt. Die involvierten Abteilungen arbeiten in klar definierten Arbeitsabfolgen mit den durch das PDM-System bereitgestellten Versionen und Releases von Produktdaten. Die

ganzheitliche Betrachtung von Systeminformationen verschiedener Domänen verliert mit fortschreitendem Produktlebenszyklus an Bedeutung. Der Fokus einer Betrachtung von modellbasierter Entwicklung von Flugzeugsystemen sollte demnach auf der Entscheidungsfindung und der Konsolidierung von Produktkonzepten liegen. Darunter fallen die Machbarkeits-, Konzept- und Definitionsphase, welche im Sprachgebrauch häufig mit *Vorentwurf* zusammengefasst werden.

2.3.2 Entwicklungsdomänen

Unabhängig von der Unternehmensstruktur existieren stets Entwicklungsabteilungen, die für die Umsetzung einer Produktidee bis hin zur Serienreife zuständig sind. Je nach Unternehmensgröße und Branche variiert der Spezialisierungsgrad dieser Abteilungen und der eingesetzten Methoden und Werkzeuge. Alle Entwicklungsabteilungen werden durch administrative Aktivitäten unterstützt und geleitet. Hierzu zählen die Projektplanung, das Projektcontrolling oder das Zulieferermanagement. Weiterhin werden Entwicklungstätigkeiten durch projektunabhängige Fertigungseinrichtungen in der Realisierung der Systemalternativen unterstützt. Hierzu zählt die Erstellung von FEM-Modellen oder Fertigungszeichnungen sowie Produktion von Prototypen oder Vorserienmodellen.

Eine Übersicht der Tätigkeiten während der Entwicklung von Flugzeugsystemen innerhalb der eben genannten Top-Level-Domänen (TLD) ist in Tabelle 2-3 dargestellt.

Tabelle 2-3 Domänen des Systementwurfs und ihre Aufgaben

TLD	Domäne	Aufgaben
Administration (unterstützend)	Projekt Management	Projektplanung (Ablaufplanung, Zielkostenmanagement, Budgetplanung, Kapazitätsplanung, …) Projektcontrolling (QG- Planung, Reviews, Berichtswesen, Meilenstein Anforderungen, …)
	Industrial Design	Form-, Farbe-, Material-, Licht-Design Konzepte Konzept-, Sales-Mockup
	Zulieferer Management	Auswahl, Monitoring von Lieferanten Integration in Entwicklungsprozess
Systementwicklung	Elektrische Systeme Mechanische Systeme Statische Systeme	Alle Tätigkeiten von Produktidee bis zur Serienreife (u.a.: Anforderungsdefinition, Funktionsdefinition, Systemspezifikationen, Simulation, Test, Verifikation und Validierung, Fertigungszeichnungen, Zertifizierung)
Fertigung (unterstützend)	Strukturdesign	FEM Analysen, Fertigungszeichnungen
	Herstellung	Herstellung von Prototypen, Vorserienmodellen, Zusammenbau und Integration

Der Fokus dieser Arbeit wird im weiteren Verlauf auf die Prozesse der Systementwicklung eingegrenzt und Schnittstellen zu Prozessen innerhalb der Administration und Fertigung werden vernachlässigt[1].

Die Komplexität von Flugzeugsystemen führt in der Realität zu Entwicklungsprozessen, die sich analog des Flugzeugentwurfsprozesses über mehrere Jahre hinweg erstrecken können. Da Flugzeugsysteme oft bei einem Zulieferer entwickelt werden und vom Flugzeughersteller nur in das Flugzeug integriert werden, arbeiten folglich mehrere Personen aus verschiedenen Fachdisziplinen zusammen. Aufgrund des Umfangs des Entwicklungsprozesses und der resultierenden Komplexität der tatsächlichen Prozessabläufe kann im Rahmen dieser Arbeit nur ein stark vereinfachter und abstrakter Entwicklungsprozess skizziert werden. Es besteht vielmehr die Zielsetzung einen repräsentativen Entwicklungsablauf bestehend aus charakteristischen Entwicklungsdomänen zu definieren, der – ohne Anspruch auf Vollständigkeit – die prinzipiellen Tätigkeiten und Herausforderungen der Entwicklung von Flugzeugsystemen verdeutlicht. Es wird demnach im Folgenden angenommen, dass am Entwurf eines Flugzeugsystems vier verschiedene Entwicklungsdomänen beteiligt sind, welche in einem sequentiellen Arbeitsablauf zusammenarbeiten (vgl. Abbildung 2-4).

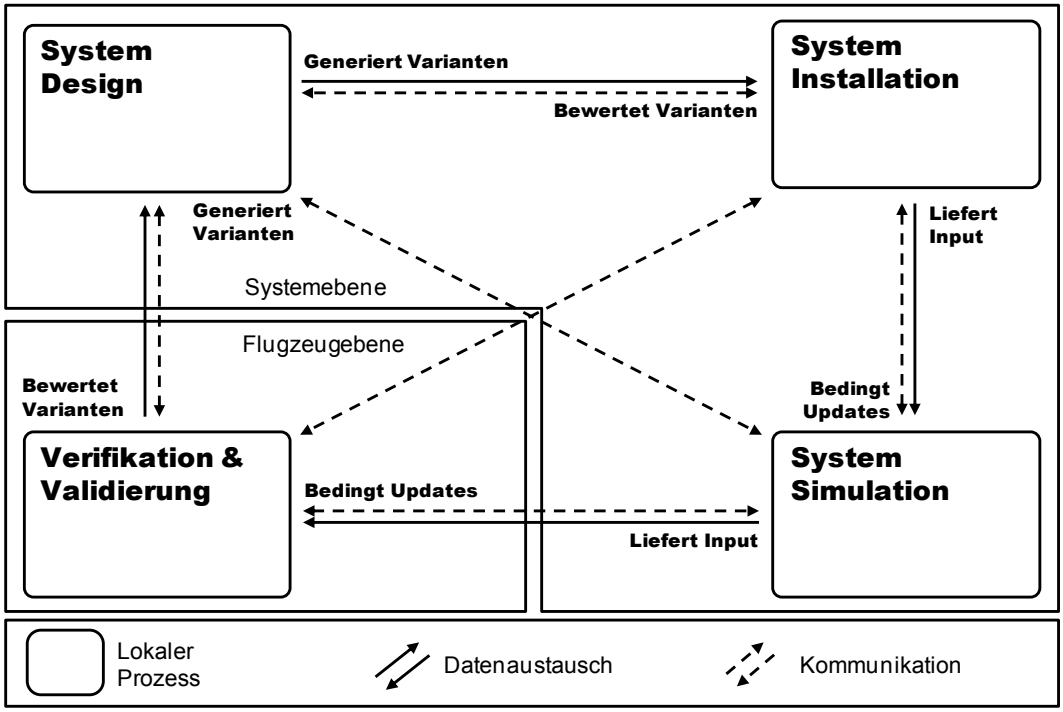


Abbildung 2-4 Domänen des Flugzeugsystementwurfs

[1] Dies schränkt den zu diskutierenden Umfang der Arbeit ein, ohne die Anwendbarkeit der entwickelten Methode im ganzheitlichen Kontext (Tabelle 2-3) zu beeinflussen (vgl. Kapitel 5).

Im Speziellen ist das *System Design* für die Synthese eines Systems zuständig: hier erfolgt das Requirements-Engineering, das Function-to-Form-Mapping bzw. ein Vergleich und die Adaption von Vorgängermodellen. Es werden die Systemfunktion und Schnittstellen zur Umgebung und entsprechende Übergabeparameter (engl.: input, output) definiert. Weiterhin werden die aus der Anforderungsvielfalt resultierenden Anwendungsszenarien (engl.: use cases) definiert. Das *System Design* erstellt die konzeptionelle Beschreibung des Systemaufbaus und generiert Konzeptalternativen.

Die Domäne *System Installation* beinhaltet Design- und Analysetätigkeiten unter Verwendung des DMU. Hierzu zählen die Zuweisung von Systemen zu bestimmten Einbauorten (engl.: space allocation), die Integration und Installation von Systemkomponenten, die Optimierung der Raumausnutzung (engl.: packaging), die Kollisionsanalyse sowie die Bereitstellung von 3D-Analysedaten.

Im Bereich *System Simulation* wird aus der funktionalen Beschreibung eines Systems ein Simulationsmodell abgeleitet. Je nach Anzahl der Anwendungsfälle, muss ein Simulationsmodell in verschiedenen Versionen vorgehalten werden. Während der Erstellung des Simulationsmodells muss parallel die Verifikation der Simulationsarchitektur erfolgen. Simulationsmodelle erfüllen verschiedene Aufgaben wie die Systemidentifikation und Parameterbestimmung oder die Optimierung von Regelgrößen.

Der Bereich *Verifikation und Validierung* vergleicht schließlich die Ergebnisse der verschiedenen Partialmodelle (funktionale Systemarchitektur, -integration, -verhalten) mit den Anforderungen und leitet notwendige Änderungsarbeiten ein. Die Systeme können dabei je nach Art des Meilensteins im Entwicklungsablauf auf Systemebene oder Gesamtsystemebene (Flugzeugebene) geprüft werden.

Wie bereits erwähnt, stellen die vier vorgestellten Entwicklungsdomänen eine vom Autor für sinnvoll erachtete Klassifizierung von Tätigkeiten innerhalb eines Entwicklungszyklus dar. Für weitere Anwendungen der vorgestellten Methode können diese Domänen angepasst oder erweitert werden. Beispielsweise könnte die Domäne *System Simulation* unterteilt werden in FEM- und CFD-Simulationstätigkeiten bzw. die Domäne *System Installation* durch die detailliertere Beschreibung von Bauraum- oder HMI-Untersuchungen aufgegliedert werden. Eine Erweiterung der Domänen um Tätigkeiten der Codegenerierung oder des physikalischen Tests ist ebenso möglich.

2.3.3 Systemsichten

Zur ausreichenden Repräsentation aller relevanten Aspekte der unterschiedlichen Fachdisziplinen, bedarf es in jedem Prozessabschnitt spezialisierter Partialmodelle, um

spezifische Aspekte eines Systems darstellen zu können. Im Regelfall wird für jedes dieser Modelle eine ebenfalls domänenspezifische Software eingesetzt. Diese Partialmodelle entsprechen dem spezifischen Bedarf der Systementwickler, einen Teilaspekt eines Systems optimal darstellen oder berechnen zu können und diese Tätigkeit mit einer auf den Entwickler zugeschnitten Softwareunterstützung auszuführen. In Anlehnung an die in Abbildung 2-4 vorgestellten Entwicklungsdomänen werden im Folgenden charakteristische Produktsichten erläutert, die während der Entwicklung von Flugzeugsystemen eine wesentliche Rolle spielen.

Konfigurationssicht: Die *Konfigurationssicht* wird von der Domäne *System Design* erstellt und beschreibt die funktionale Architektur eines Systems. Es werden die Systemelemente hierarchisch beschrieben und die Beziehungen zwischen den einzelnen Elementen dargestellt. Die Beziehungen können dabei physikalischer oder logischer Art sein. Zur Erstellung der Modelle der Konfigurationssicht werden üblicherweise Office-Anwendungen wie Microsoft Office[2] in Kombination mit Anforderungsmanagementwerkzeugen wie Rational Doors[3] eingesetzt.

Konstruktionssicht: Die konstruktive Sicht beschreibt die geometrische Gestalt der Elemente sowie deren geometrischen Zusammenhänge. Diese können je nach Entwicklungsfortschritt 2D-Skizzen oder 3D-Volumenkörper sein und werden in kommerziellen Luftfahrtbetrieben mit CAD-Systemen wie CATIA V5[4], NX Unigraphics[5] oder Pro/Engineer[6] erstellt. Die konstruktive Sicht setzt die Vorgaben der *Konfigurationssicht* für detaillierte Analysen der Integration, Statik oder Kinematik in ein CAD-Modell um. Abbildung 2-5 zeigt das dreidimensionale Modell eines Verkehrsflugzeuges, welches unter anderem für Integrationsanalysen von Systemen oder als graphische Diskussionsgrundlage herangezogen werden kann.

Abbildung 2-5 CAD-Modell eines Verkehrsflugzeugs

[2] www.office.microsoft.com
[3] www.ibm.com/software/products/de/de/ratidoor
[4] www.3ds.com/products/catia
[5] www.plm.automation.siemens.com
[6] www.ptc.com/product/creo (Pro/Engineer ist seit 2011 unter dem Namen *Creo* bekannt)

Simulationssicht: Wie bereits angedeutet, dient die *Konstruktionssicht* häufig als Basis für die Simulation spezifischer Bereiche eines Systems. Ausgehend von der Systemgeometrie werden innerhalb der Modelle der *Simulationssicht* beispielsweise die Thermodynamik oder Aerodynamik auf Basis numerischer Verfahren bestimmt. Die zumeist sehr genauen CAD-Modelle werden zum Zwecke der Simulation wieder vereinfacht, um möglichst ressourcenschonend die rechenintensiven Simulationen durchführen zu können. Weiterhin dienen Simulationsmodelle zur Optimierung von Systemparametern oder zur Co-Simulation, bei der verschiedene Modellierungswerkzeuge und Solver in einem Simulationsablauf gekoppelt werden.

Dokumentationssicht: Der Bereich der Verifikation und Validierung steht stellvertretend für die integrative Betrachtung der Systementwicklung und die Absicherung der Systemfunktion. Dabei wird nur ein bruchteilhafter Ausschnitt aller Systemparameter berücksichtigt, der sich oftmals lediglich auf eine Kennzahl reduziert und zum Anforderungsabgleich herangezogen wird. Damit eng verknüpft ist die Aufgabe, die im Laufe eines Entwicklungszyklus entstehenden Versionen und Varianten eines Systems freizugeben und zu verwalten. Die *Dokumentationssicht* sorgt für die kontinuierliche Archivierung der Entwicklungsergebnisse und schafft damit einerseits die Basis für die Zulassung des entwickelten Systems durch die Luftfahrtbehörden und andererseits für die Wiederverwendung der Daten bei Neuentwicklungen.

2.4 Defizite der aktuellen Systementwicklungsprozesse

2.4.1 Unternehmensstruktur und –kultur

Laut VILSMEIER (2011) existieren Schwierigkeiten, die Potenziale neuer Methoden und Werkzeuge auszuschöpfen, da etablierte Verfahren nur schwer zu ändern sind und Schnittstellen zur Integration neuer Werkzeuge nicht definiert sind. Oftmals fehlen klar definierte Arbeitsabläufe (engl.: workflow), die den Prozess der Datenerzeugung und Datenbereitstellung zwischen Abteilungen definieren. Gerade für die Entwicklung von Multi-Domänen Produkten ist ein Vorgehensmodell für die Beschleunigung der Entwicklungsabläufe notwendig. Doch die exakten Wechselwirkungen von Fachabteilungen werden im derzeitigen Entwicklungsprozess von Flugzeugsystemen kaum berücksichtigt (LANGERMANN, 2009). Der Informationsaustausch wird zum einen durch unterschiedliche Berechnungswerkzeuge und deren Datenformate behindert. Zum andern bestehen domänenspezifische Begriffswelten, Methoden und Verfahren sowie unterschiedliche Ausbildungsniveaus (LANGERMANN, 2009; SHEA, 2011). Die resultierenden Teilentwicklungen einzelner Abteilungen sind untereinander inkompatibel

und im schlimmsten Fall inkonsistent. Das kann zu zeit- und kostenintensiven Iterationszyklen im Entwicklungsprozess führen (KRAUSE, 2007). Ein weiterer Grund für Schnittstellenprobleme zwischen Entwicklungsabteilungen kann die Angst vor Wissensverlust an nachfolgende Prozesspartner sein (LANGERMANN, 2009).

Die Erzeugung oder Auswahl neuer Methoden der Systementwicklung sollte sich deshalb an bestehenden Verfahren orientieren und die Schnittstellen einer möglichen Implementierung berücksichtigen. Desweiteren sind die Barrieren zwischen Abteilungen zu identifizieren und in klar definierte Schnittstellen umzuwandeln. Auch das Systemverständnis der am Prozess beteiligten Akteure ist von entscheidender Bedeutung (SHEA, 2011). Airbus führte aus diesem Grund verschiedene Konzepte zur Unterstützung des parallelen Entwickelns (engl.: Concurrent Engineering; vgl. Kapitel 3.3.1) ein. Der Leitgedanke „think process, then method and then tool“ soll die Ingenieure dazu animieren, die Schnittstellen ihrer Arbeitsumfänge kennenzulernen und die Prozessorganisation soweit zu berücksichtigen, dass Arbeitsabläufe schlank gehalten werden können und gleichzeitig die Produktqualität maximiert wird (PARDESSUS, 2004).

2.4.2 Daten- und Wissensbasis

Es werden heute verschiedene Formen der digitalen Daten- und Wissensspeicherung angewendet. Die einfachste Form der Datenverwaltung stellen Netzlaufwerke dar, welche aber große Einschränkungen bei gleichzeitiger Benutzung mehrerer Anwender mit sich bringen. Die meist genutzte Form ist die Verwaltung von Daten durch Produktdatenmanagement-Systeme (PDM) aufgrund ihrer erhöhten Funktionalität. Ein PDM-System dient als Integrationsplattform für die entstehenden Produktdaten und der Steuerung von Entwicklungsabläufen (ANDERL, 2011). Der verstärkte Einsatz von CAx-Systemen und der damit verbundene Organisationsaufwand einer Vielzahl von digitalen Daten hat den Einsatz von PDM- Systemen dabei erst notwendig gemacht. PDM-Systeme greifen je nach Umfang und Art der Daten auf relationale, objektrelationale oder objektorientierte Datenbanken (RDB, ORDB, OODB) zurück, welche über mehrere Standorte verteilt sein können (DDB) (VOSSEN, 2008; SHEA, 2011).

LANGERMANN (2009) beschreibt den Einsatz von anwendungsbezogenen Datenbanken bei Airbus: Der Austausch von Geometriedaten erfolgt über die VPM-Datenbank (Virtual Product Management); Anforderungsdokumente werden im EDMS (Engineering Data Management System), große Simulationsdatenmengen im SimDMS verwaltet und standortübergreifende Daten werden in der AMDB (Aircraft Model Data Base) gelagert. Das Ziel des gesicherten und redundanzfreien Modellaustausches zwischen Entwicklungsabteilungen wird aber oftmals nicht erreicht. In vielen Fällen werden

aufgrund von Unübersichtlichkeit wiederholt Grunddaten erzeugt, statt bereits bestehende Lösungen zu nutzen (ANDERL, 2011). Durch die komplexe Zugriffsverwaltung der PDM-Systeme werden zudem Modelle dezentral bearbeitet und gespeichert, wodurch die bereichsübergreifende Konsistenz kompromittiert wird (LANGERMANN, 2009). Weiterhin besteht noch ein großer Teil an dokumentenbasierten Arbeitsabläufen, wodurch die Möglichkeit der Verwendung von veralteten Dokumenten entsteht. Eine zentrale Datenbank und die einheitliche Benutzung digitaler Dokumente verhindern diese Problematik.

PARDESSUS (2004) beschreibt das „Multi-View-DMU“ als zukünftige, zentrale Datenbasis, welche durch ein PDM-System verwaltet wird und damit als zentraler Datenbestand für alle Entwicklungsdomänen zur Verfügung steht. Die Herausforderung stellen dabei die verbesserte Integration und Unterstützung der Design- und System Engineering-Tätigkeiten im Entwurf des virtuellen Flugzeugs dar. Der aktuelle Mangel an Werkzeugunterstützung der Synthese-, Konzept- und Designphase wird auch in ANDERL (2011) und SHEA (2011) beschrieben.

2.4.3 Werkzeugvielfalt und -integration

Zur Definition und Erstellung komplexer technischer Produkte ist eine Vielzahl von Werkzeugen notwendig, die spezifische Aufgaben im Kontext spezifischer Theorien und Methoden erfüllen. Diese Werkzeuge sind für eine konkrete Aufgabe entwickelt worden und nicht für die Interaktion mit anderen Werkzeugen (ANDERL, 2011). Um den Informationsfluss zwischen verschiedenen Abteilungen herzustellen, werden von den Ingenieuren heute „Ad-hoc“-Lösungen geschaffen, die keinem regulierten Prozess unterliegen (BROY ET AL., 2010). Durch das Fehlen eines koordinierten Prozesses bleibt das Wissen der Ingenieure über Produktdatentransfer nur implizit abrufbar und die verwendeten Werkzeuge bleiben weitestgehend autonom (Abbildung 2-6).

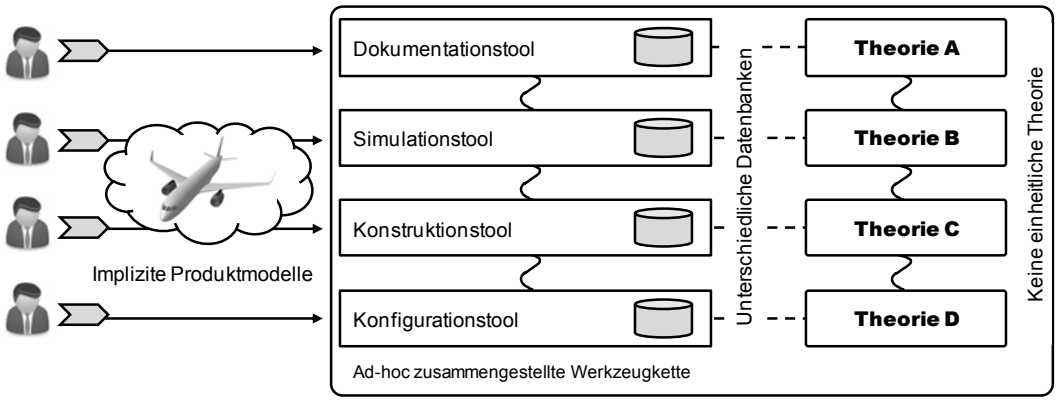


Abbildung 2-6 Heutige Entwicklungsumgebung
Quelle: BROY ET AL. (2010), leicht modifiziert

Um die Kommunikation zwischen zwei Softwarewerkzeugen herzustellen, existieren zwei grundsätzliche Ansätze. Zum einen besteht die Möglichkeit der systemspezifischen Schnittstellengestaltung, bei der zwischen *n* Softwarewerkzeugen *n(n-1)* Konverter zur Übersetzung zwischen Quell- und Zielformaten benötigt werden. Zum anderen existiert der Austausch über systemneutrale Schnittstellen, bei der alle beteiligten Softwarewerkzeuge ein neutrales Datenformat interpretieren können und die Anzahl der benötigten Konverter auf *2n* Konverter beschränken (vgl. ANDERL (2011), SHEA (2011)). Vor allem im Bereich der CAD-Systeme hat sich eine Vielzahl von neutralen Datenformaten etabliert (vgl. Tabelle 2-4), die sich hauptsächlich durch den abzubildenden Informationsgehalt unterscheiden. Dabei ist festzuhalten, dass der Umfang der austauschbaren Daten mittels neutraler CAD-Datenformate sowohl von der Fähigkeit des neutralen Datentyps abhängt, als auch von den Kapazitäten des empfangenden Zielsystems. Tabelle 2-4 zeigt deutlich, dass die meisten neutralen Datenformate auch nichtgeometrische Daten speichern können, der Umfang ist jedoch stark unterschiedlich und für viele interdisziplinäre Anwendungen nicht ausreichend. Weiterhin besteht die Problematik, dass bei Einsatz eines neutralen Datenformates alle am Entwurf beteiligten Werkzeuge dieses Format zu lesen im Stande sein müssen. Dies ist beispielweise im Rahmen einer gemeinschaftlichen Entwicklung von CAD-Bauteilen zwischen OEM und Zulieferer der Fall. Hier werden neutrale Datenformate benutzt, um beispielsweise ausschließlich Geometriedaten (z.B.: 3D-Flächenmodelle) auszutauschen und weitere Information (z.B.: Ingenieurwissen in Form von parametrisierten Oberflächen) geheim zu halten.

Tabelle 2-4 Eigenschaften neutraler CAD-Datenformate
Quelle: STOYE, (2011), leicht modifiziert

Eigenschaft	IGES	STEP	JT	DXF	STL	VRML
Norm (n) / Industriestandard (i)	n	n	i	i	i	n
ASCII (a) / Binär (b) / XML (x)	a + b	a + x	b	a	a + b	a
Inhalt						
2D-Modelle	x	x	–	x	–	–
3D-Modelle (Flächen)	x	x	x	–	x	x
3D-Modelle (parametrische Solids)	x	x	x	x	–	–
Nichtgeometrische Daten	x	x	x	x	–	x

Im Vergleich dazu kommen innerhalb interdisziplinärer Entwicklungsaufgaben mechatronischer Produkte verschiedenste Werkzeuge zum Einsatz, die in den wenigsten Fällen eine Import-/ Exportfunktionalität für CAD-basierte, neutrale Datenformate besitzen.

Ein weiteres Defizit der aktuellen Werkzeuglandschaft stellt der Funktionalitätsumfang der am Markt vorhandenen Werkzeuge dar. Dieser ist meist derart groß, dass die

praktische Anwendung der Software behindert wird. Die Art der Anwendung des gleichen Werkzeuges ist oft stark unterschiedlich, es werden Funktionsumfänge nur bruchteilhaft genutzt (VILSMEIER, 2011). Die Forderung nach einem einheitlichen Werkzeug, das alle Tätigkeiten im Entwicklungsprozess bei Hersteller und Zulieferer gleichermaßen gut erfüllt, ist demnach utopisch. Die Verwendung einer Vielzahl von Werkzeugen mit spezialisiertem Funktionsumfang entspricht der Forderung nach Werkzeugen, die nicht überladen sind und die Gebrauchstauglichkeit einschränken. Folglich ist eine Zunahme der Werkzeugvielfalt eher als Prämisse innerhalb der Zukunft der Systementwicklung zu verstehen. Die Möglichkeit zur bereichsübergreifenden Werkzeugintegration muss unter dieser Prämisse nicht zwangsläufig leiden, denn die zunehmende Verwendung von einheitlichen Programmiersprachen und Open Source Software kann die Schnittstellengestaltung sogar erleichtern.

2.5 Identifikation des Handlungsbedarfes

Die Zukunft der Systementwicklung benötigt nach BROY ET AL. (2010) drei Zutaten: 1) eine verständliche Modellierungstheorie, welche die Semantik für die formale Definition von Modellen liefert; 2) ein Produkt- und Prozessmodell, welches den strukturellen Aufbau des Produktes und der zugehörigen Prozesse wiederspiegelt und 3) eine integrierte Entwicklungsumgebung, welche die nahtlose Integration der Entwicklungswerkzeuge zulässt. Abbildung 2-7 zeigt das Schema einer modellbasierten, integrierten Entwicklungsumgebung. Verschiedene Entwickler benutzen eine gemeinsame Datenquelle, die jedem Nutzer eine spezifische Sicht auf das Modell ermöglicht. Die Arbeitsvorgänge werden durch eine Arbeitsablaufdefinition in prozeduraler Hinsicht sowie durch eine Modellierungstheorie in deklarativer Hinsicht unterstützt.

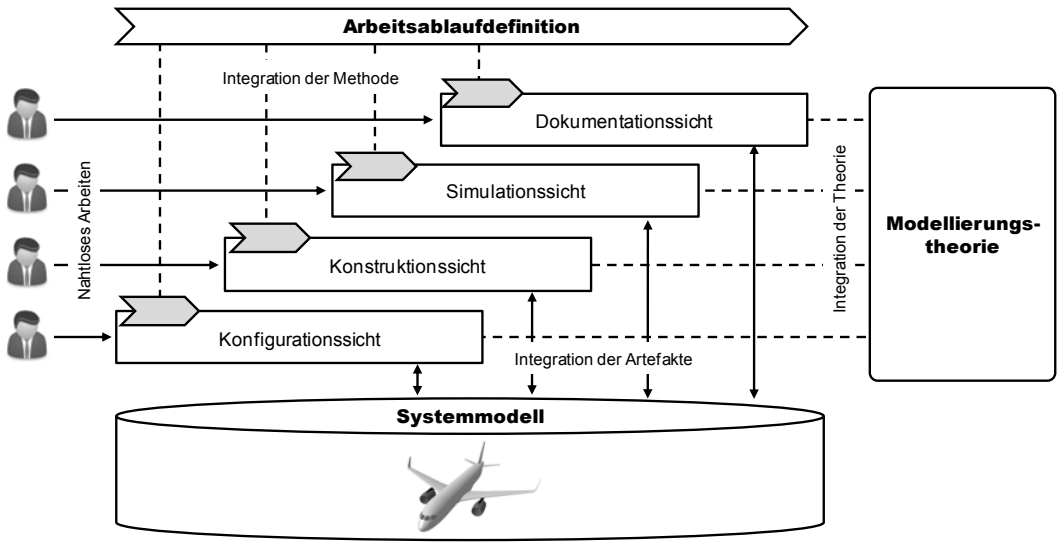


Abbildung 2-7 Modellbasierte integrierte Entwicklungsumgebung
Quelle: BROY ET AL. (2010), leicht modifiziert

Das Vorgehensmodell nach BROY ET AL. (2010) oder ähnliche Modelle einer zukünftigen Entwicklungsumgebung finden sich vielfältig in der Literatur (vgl. SJÖSTEDT, 2009; MAUL, 2009; QAMAR, 2011). Die Realisierung solcher Artefakte (hier: Produktmodelle) ist Gegenstand aktueller Forschung und wird vor allem durch die Untersuchung adäquater Modellierungssprachen und die Umsetzbarkeit eines branchenunabhängigen Produktmodells beherrscht.

Das größte Problem bei der praktischen Umsetzung modellbasierter Vorgehen ist die Anforderungsvielfalt, die aus Unternehmensspezifika und der bestehenden Werkzeugvielfalt resultiert. Weiterhin sind die vielfältigen Ansprüche der potenziellen Nutzer eines Metamodells zu erfassen, die vor der Implementierung berücksichtigt werden müssen. Eine ungenaue Anforderungsdefinition kann v.a. in späten Phasen der Entwicklung zu unzureichender Funktionalität der Werkzeuge führen. Weiterhin beschreibt VILSMEIER (2011), dass die Implementierung einer neuen Methode oder eines neuen Werkzeuges an der vorhandenen Produkt- oder Prozesskomplexität eines Unternehmens scheitern kann. Dieser Umstand kann nur durch die passende Definition des Abstraktionsgrades der zu modellierenden Systeme und Prozesse behoben werden.

Tabelle 2-5 fasst die Defizite zusammen, die innerhalb der Systementwicklung über verschiedene Fachgebiete hinweg identifiziert worden sind. Aktuelle Defizite sollten als künftige Herausforderung formuliert werden und bei der Betrachtung neuer Entwurfsansätze besondere Beachtung finden.

Tabelle 2-5 Defizite der aktuellen Systementwicklung

			Autor															
		Fokus: Flugzeuge / Luftfahrtsysteme	✓	✓	✓	✓												
		Fokus: Mechanische Produkte					✓	✓	✓	✓								
		Fokus: Mechatronik									✓	✓	✓	✓				
		Fokus: Software- / Systemsengineering													✓	✓	✓	✓
Kategorie	**#**	**Defizite**	Moir und Seabridge, 2003	Vermeulen, 2007	Langermann, 2009	Kirby, 2001	Brandstätter, 2009	Shea, 2011	Vilsmeier, 2011	Anderl, 2011	Buur, 1990	Isermann, 2008	Sjöstedt, 2009	Qamar, 2011	Günzler, 2005	Broy, 2010	Weilkiens, 2008	Friedrich, 2010
System	1	Entwicklungskomplexität mechatronischer Systeme	✓				✓				✓	✓	✓	✓				
	2	Redundanz von (analogen) Systemdaten			✓			✓	✓	✓			✓	✓				
	3	Fehlende Abbildungsmöglichkeit verschiedenartiger Systemrelationen			✓						✓				✓		✓	✓
	4	Mangelnde Verfügbarkeit ganzheitlicher Systeminformation			✓	✓	✓			✓			✓			✓		
	5	Fehlende Darstellung verschiedener Sichten auf ein System						✓		✓					✓	✓		
Systementwicklung	6	Verfügbarkeit einer Modellierungsmethode und zugehöriger Semantik		✓		✓		✓		✓	✓				✓	✓	✓	
	7	Mangelnde Verwendung von wissensbasierten Methoden		✓			✓	✓		✓			✓		✓			
	8	Integriertes Anforderungs- und Funktionsmanagement				✓	✓	✓		✓	✓							✓
	9	Geringer Integrationsgrad der Systementwicklungsdomänen	✓	✓	✓	✓	✓		✓				✓	✓	✓	✓		✓
	10	Adequate Toolunterstützung der Modellbildung		✓	✓	✓	✓									✓	✓	
	11	Verfügbarkeit / Akzeptanz adäquater Modellierungssprache		✓		✓	✓	✓		✓			✓	✓	✓	✓	✓	✓
	12	Systemverständnis der beteiligten Akteure					✓	✓		✓		✓						
	13	Kompatibilität verwendeter Software / Modelle		✓	✓		✓		✓			✓	✓	✓		✓		
	14	Produktdatenmanagement von Multidomain- Produkten					✓	✓	✓			✓			✓	✓		

Zusammenfassend lässt sich feststellen, dass in der aktuellen, rechnergestützten Entwicklung komplexer Flugzeugsysteme mehrere, verschiedene Partialmodelle des gleichen Systems existieren. Die Gründe hierfür liegen in dem unterschiedlichen Systemverständnis der beteiligten Personen und Abteilungen, dem Zweck der Modellbildung oder den Möglichkeiten der angewendeten Softwarewerkzeuge. Die erstellten Partialmodelle werden jedoch meist isoliert voneinander verwendet und sind aufgrund der Modellierungsansätze und -sprachen untereinander inkompatibel. Nachdem aber ein gewisser Datenaustausch zwischen Domänen im iterativen Entwurfsablauf notwendig ist, werden manuell aufwändige Schnittstellen geschaffen, die oft nur einen spezifischen Zweck erfüllen und nicht dauerhaften Nutzen bringen (vgl. Kapitel 2.4.3).

Als Folge der beschriebenen Problematik aktueller Entwicklungsprozesse, lässt sich der Wunsch formulieren, einen konzeptionellen Rahmen zu schaffen, der die Kommunikation zwischen Partialmodellen ermöglicht und damit die Querbezüge zwischen Produktsichten und Modellen explizit ermöglicht. Zu diesem Zweck ist ein Metamodell notwendig, das domänenübergreifende Informationen aus einzelnen Partialmodellen speichern kann und diese in einem globalen Kontext wieder zur Verfügung stellt. Dazu sind die Anforderungen an ein solches Metamodell zu identifizieren und die aktuellen Entwurfsmethoden und Werkzeuge zu analysieren, die aktuell in der Luft- und Raumfahrt Anwendung finden. Das zu entwerfende Metamodell sollte flexibel an unterschiedliche Problemstellungen angepasst und in bestehende Werkzeugketten integriert werden können, um eine erfolgreiche Implementierung und nachhaltige Anwendbarkeit zu gewährleisten. Auf diese Aspekte wird in den folgenden Kapiteln im Detail eingegangen.

3 METHODEN DER SYSTEMENTWICKLUNG

Im folgenden Kapitel werden aktuelle Ansätze und Richtlinien der Systementwicklung diskutiert und daraus Potenziale für die Entwicklung von Flugzeugsystemen abgeleitet. Es sei angemerkt, dass für eine ganzheitliche und integrative Betrachtung von Systemen stets eine Betrachtung der Systeme selbst aber auch der zugrundeliegenden Prozesse und der umgebenden Organisation wichtig ist. Nachfolgend werden aktuelle Methoden der Systementwicklung analysiert und aus den Ergebnissen dezidierte Handlungsempfehlungen für den effizienten Entwurf von Flugzeugsystemen abgeleitet.

BUUR (1990) erkennt, dass klassische Produktentwicklungsansätze allein nicht ausreichen, um erfolgreich mechatronische Systeme zu entwickeln. Es bedarf vielmehr der Kombination und Koordination von Prozessen und Werkzeugen aus den Bereichen Mechanik, Elektronik und Softwareentwicklung. Um eine Übersicht über aktuelle Handlungsempfehlungen, Entwicklungsmethoden und Richtlinien innerhalb der genannten Bereiche zu erlangen, wird die von HUBKA (1990) beschriebene Struktur von Entwicklungswissenschaften angewandt. Er unterscheidet zwischen deskriptiven (beschreibenden) und präskriptiven (vorschreibenden) Verfahren sowie zwischen Verfahren mit Fokus auf das technische Produkt bzw. Fokus auf den (Konstruktions-) Prozess. Abbildung 3-1 zeigt die Beeinflussung des modellbasierten Systementwurfs durch eine Vielzahl dieser Handlungsempfehlungen, Entwicklungsmethoden und Richtlinien. Die Darstellung erfolgt in Anlehnung an BLESSING und CHAKRABARTI (2009) in einem sogenannten ARC-Graphen (engl.: Areas of Relevance and Contribution). Die grau eingefärbten Bereiche zeigen Fachgebiete, welche eine besonders hohe Relevanz für den Kontext dieser Arbeit besitzen und in den folgenden Kapiteln diskutiert werden. Die weißen Bereiche sind thematisch weniger relevant, werden aber in Ihren Grundsätzen berücksichtigt. Eine detailierte Diskussion dieser Bereiche würde den Rahmen dieser Arbeit sprengen.

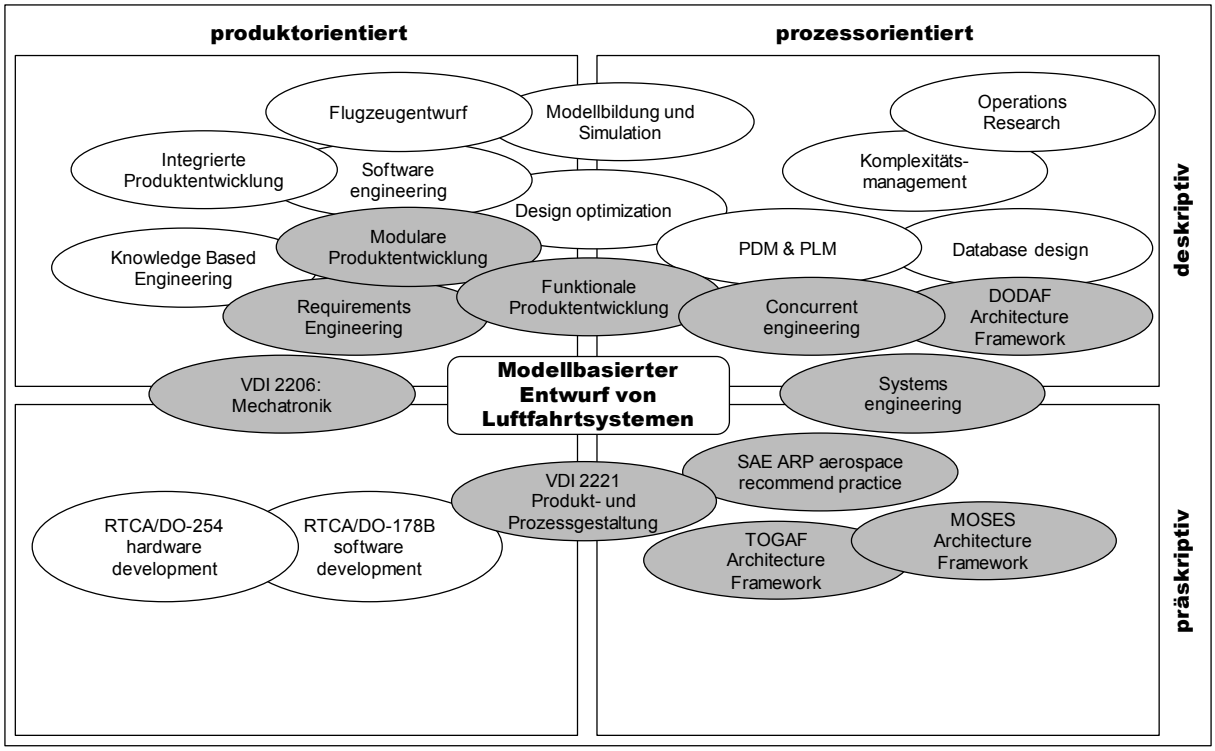


Abbildung 3-1 Relevante Fachgebiete zur modellbasierten Entwicklung (Darstellung in Anlehnung an HUBKA, 1990; BLESSING und CHAKRABARTI, 2009)

3.1 Deskriptive Produktentwicklungsansätze

3.1.1 Requirements Engineering

Der Begriff *Requirements Engineering* (RE) bezeichnet eine „iterative, systematische, auf Effizienz und Effektivität ausgelegte Vorgehensweise mit dem Ziel, eine explizite, mit allen Stakeholdern abgestimmte Anforderungs- und Systemspezifikation zu erstellen“ (BROY, 2011). Diese Vorgehensweise besteht aus folgenden Teilschritten:

- Anforderungen erheben (engl.: elicitation)
- Anforderungen analysieren, verhandeln, konsolidieren (engl.: analysis)
- Anforderungen strukturieren, modellieren, dokumentieren (engl.: specification)
- Anforderungen auf Qualität und Gültigkeit überprüfen (engl.: validation & verification)

Das Finden, Bewerten, Dokumentieren, Strukturieren und Verwalten von Anforderungen (MAIER UND MATTHES, 2011) innerhalb eines RE-Prozesses kann sowohl für Prozesse, Produkte, Software-intensive-Systems (SIS) sowie Software eingesetzt werden. RE ist, wie bereits in Kapitel 2.3 beschrieben, ein Teil des V-Modell und damit eine etablierte Grundlage für die Entwicklung von luftfahrttechnischen Systemen und Flugzeugen. Ein Grund für ein stringentes Anforderungsmanagement liegt in der Zulassung

der Systeme (ARP4754A, 2010) sowie einer besseren Grundlage zur Strukturierung von Entwicklungsprozessen.

Eine Anforderung kann als Eigenschaft verstanden werden, die ein Stakeholder fordert und die vertraglich festgehalten wird (MAIER UND MATTHES, 2011). Eine Anforderung beschreibt lediglich, was für eine Eigenschaft dargestellt werden soll, nicht wie diese Eigenschaft erreicht werden soll. Diese Aufgabe obliegt dem Systemhersteller und wird im Rahmen der Funktionalen Produktentwicklung (Kapitel 3.1.2) umgesetzt. Um die meist große Vielzahl an Anforderungen handhabbar zu machen, empfiehlt sich eine Einteilung in Hierarchien. In der Entwicklung von Airbus Flugzeugen ist beispielhaft folgende Einteilung zu finden (DOUMBIA; LAURENT; ROBACH & DELAUNAY, 2010):

Tabelle 3-1 Anforderungsklassifikation im Flugzeugentwurf

Ebene	Kurzzeichen	Dokument	Beschreibung
Flugzeugebene	L0 (Level 0)	Top Level Aircraft Requirements Document (TLARD)	Anforderungen auf Programmebene
Systemebene	L1 (Level 1)	System Requirement Document (SRD)	Anforderungen auf Systemebene
Subsystemebene	L2 (Level 2)	Detailed Functional Specification (DFS)	Anforderungen auf Subsystemebene
Ausrüstungsebene	L3 & L4 (Level 3 & 4)	Test Requirements Document (TRD) & SCADE	DFS- Testspezifikationen & SCADE Modell

Diese Einteilung entspricht der Struktur der ATA-Kapitel, welche eine Systematik für die Gliederung von Flugzeugsystemen für zivile Transportflugzeuge der Zulassungsklasse FAR-25 bzw. CS-25 darstellen. Anforderungsverwaltung wird meist durch ein Werkzeug unterstützt, welches typischerweise durch ein Datenbanksystem realisiert wird, welches über eine graphische Benutzeroberfläche das interaktive Darstellen, Sortieren und Auswerten von Anforderungen ermöglicht. Anforderungen können mit Attributen versehen werden, was eine weitere Unterteilung und verschiedene Sichten der Anforderungen zulässt. Ein Beispiele für die Attributierung von Anforderungen ist die Zuordnung zu funktionalen und nicht-funktionalen Anforderungen oder die Zuordnung zu Anforderungsklassen (ökonomisch, ökologisch, prozedural, u.a.).

3.1.2 Funktionale Produktentwicklung

Die Theorie der *Funktionalen Produktentwicklung* ergänzt und vervollständigt die Ansätze des *Requirement Engineering*. Anforderungen stellen v.a. zu Beginn einer Entwicklung abstrakte Formulierungen dar, die in konkrete Funktionen übersetzt werden müssen. Ein Ziel der funktionalen Produktbeschreibung ist die Reduktion der Komplexität der konstruktiven Aufgabenstellung durch die Strukturierung des Gesamtsystems in einfachere Teilsysteme (MEIER, 2005, S. 334). Weiterhin erlaubt die funktionale Beschreibung eine Analyse gleichartiger Funktionen und die Identifikation von

Modulen. Modulare Entwicklungsumfänge können analog der Systemtheorie als Blackbox betrachtet werden (WALTER, 2010) und erlauben die unabhängige Entwicklung (Concurrent Engineering, siehe Kapitel 3.3.1) oder die Fremdvergabe der Entwicklung (Outsourcing).

Die *Funktionale Produktentwicklung* läuft in Anlehnung an WEBER ET AL. (2002) in drei Schritten ab. In der *Synthese* werden aus den geforderten Eigenschaften (Anforderungen) die zu erfüllenden Eigenschaften (Funktionen) abgeleitet, die während der *Analyse* auf Basis von Systemmodellen identifiziert werden. Schließlich erfolgen eine *Evaluierung* der Ergebnisse und die resultierende Festlegung von (interdisziplinären) Maßnahmen, mit denen Abweichungen vom Soll-Zustand behoben werden können. Abbildung 3-2 zeigt den Prozess der *Funktionalen Produktentwicklung* und die Notwendigkeit der expliziten Verknüpfung von Anforderung und Funktion (engl.: mapping), um erfolgreich anforderungsgerecht zu entwickeln. Der Stellenwert dieser Verknüpfung und die Notwendigkeit einer entsprechenden Abbildung innerhalb der Methoden und Werkzeuge werden u.a. in BRANDSTÄTTER (2009) UND FRIEDRICH (2010) erwähnt.

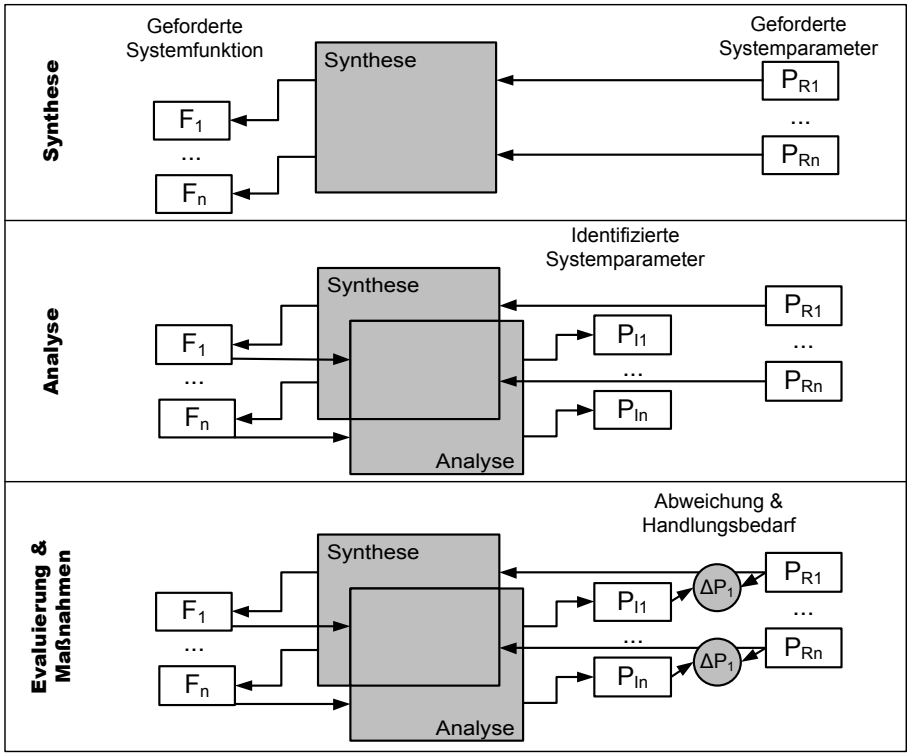


Abbildung 3-2 Funktionale Produktentwicklung
Quelle: WEBER ET AL. (2002, S.106); leicht modifiziert

3.1.3 Modulare Produktentwicklung

Eine Spezialisierung der *Funktionalen Produktentwicklung* stellt die modulare Entwicklung von technischen Produkten dar. Die Modularisierung eines Produktes beschreibt die Zerlegung eines Produktes in substitutionsfähige, austauschbare Teile oder Module. Dies wird möglich durch die Funktionsanalyse des Gesamtproduktes und der Identifikation von Baugruppen, die klar abgegrenzte Funktionsumfänge erfüllen. Module zeichnen sich folglich durch klar definierte Schnittstellen aus und werden maßgeblich durch ihre Funktionen bestimmt. Der Hauptvorteil von modular gestalteten Produkten ist die Unterteilung in überschaubare Einheiten, die unabhängig voneinander entwickelt, konstruiert und beschafft oder produziert werden können. Die Komplexität im Unternehmen kann durch die entstehende Reduktion von Verknüpfungen deutlich vermindert werden. Weiterhin werden Montage-, Aktualisierungs- oder Wartungsarbeiten stark vereinfacht, flexibilisiert und beschleunigt (PICOT und BAUMANN, 2007).

Bei der Entwicklung von modular aufgebauten Produkten müssen zunächst die einzelnen Funktionen des Gesamtsystems in Analogie zur Funktionalen Produktentwicklung definiert werden. Hierbei lassen sich zwei grundlegende Bauprinzipien voneinander abgrenzen, die auch jeweilige Mischformen bilden können. Die Modulbauweise und die Integralbauweise von Produkten stehen im Gegensatz zueinander. Diese beiden Bauprinzipien lassen sich anhand zweier Eigenschaften voneinander abgrenzen. Modulare Produktarchitekturen erfüllen das Kriterium der funktionalen Unabhängigkeit (GÖPFERT, 2009, S.104). Dies bedeutet, dass bei einem modular aufgebauten Produkt eine Komponente genau eine Teilfunktion des Gesamtsystems erfüllt. Bei der Integralbauweise hingegen integriert ein Bauteil mehrere Funktionen. Ein weiteres Unterscheidungsmerkmal ist die Entkoppelung von Schnittstellen. Je stärker die Schnittstellen voneinander losgelöst sind, desto modularer stellt sich die Produktarchitektur dar und desto eigenständiger können die verschiedenen Module im Entwicklungsprozess behandelt werden (KERSTEN, 2002). Dieses Prinzip der physischen Unabhängigkeit erfüllen modulare Produkte, da sie aufgrund ihrer standardisierten Schnittstellen leicht voneinander trennbar sind. Mit der Zunahme der beiden oben beschriebenen Eigenschaften steigt die Modularität von Produkten.

Integral aufgebaute Produkte zeichnen sich hingegen dadurch aus, dass ihre Komponenten physisch nicht voneinander trennbar sind und eine mehrdeutige Zuordnung zwischen ihnen und ihren Funktionen besteht. Dadurch ergeben sich verschiedene Vorzüge und Nachteile der Integralbauweise gegenüber der Modulbauweise. Da bei

der integralen Produktarchitektur mehrere Funktionen vereint und Redundanzen vermieden werden, werden somit der Bauraum und das verwendete Material optimal ausgenutzt. Durch diese geometrisch verschachtelte Bauweise kann eine Gewichtsminimierung der Produkte erreicht werden (KERSTEN, 2002). Gerade in der Luftfahrtindustrie ist die Gewichtsersparnis von großer Bedeutung. Diesem Vorteil steht jedoch der entscheidende Nachteil gegenüber, dass integral aufgebaute Produkte im Gegensatz zu modular gestalteten Produkten über weniger standardisierte Schnittstellen und Komponenten verfügen. Somit ist bei einer modularen Produktarchitektur eine „bessere funktionsspezifische Anpassbarkeit der einzelnen Module durch gezielte Optimierung hinsichtlich nur einer Funktion“ (KERSTEN, 2002, S.69) zu erreichen. Die einzelnen Module werden jedoch oftmals für Varianten unterschiedlicher Leistungsklassen standardisiert. Dies kann zu einer Überdimensionierung der Module führen, da diese für die jeweils höchste Leistungsstufe ausgelegt werden müssen. Die Überdimensionierung ist oftmals mit einer nicht notwendigen Funktionsübererfüllung verbunden (WOHLGEMUTH - SCHÖLLER 1999).

Eine detailliertere Diskussion der Vor- und Nachteile der Modularisierung von Kabinenstrukturen ziviler Passagierflugzeuge sowie ein Umsetzungsbeispiel von modularen Kabinenstrukturen sind in MOHR (2010) zu finden. Das Prinzip der modularen Systementwicklung findet sich jedoch nicht nur in der Gestaltung physischer Produkte sondern auch im Bereich der Softwareentwicklung. Deshalb spielen innerhalb der Entwicklung mechatronischer Systeme modulare Gestaltungsprinzipien und Methodeneinsatz eine zunehmend große Rolle.

3.2 Präskriptive Produktentwicklungsansätze

3.2.1 Entwicklungsmethodik für technische Systeme

Die VDI Richtlinie 2221 beschreibt eine allgemeine und branchenunabhängige Vorgehensweise zur methodischen Entwicklung von technischen Systemen. Die Richtlinie stellt eine Leitlinie für die Abfolge von Arbeitsabschnitten und deren notwendigen Ergebnissen dar. Unter Verwendung einer klaren Nomenklatur sowie konkreter Methodenvorschläge und Praxisbeispiele wird ein Vorgehensmodell für den Entwicklungs- und Konstruktionsprozess technischer Produkte vorgegeben. Die Richtlinie stellt das konsolidierte Ergebnis aus Industrieprozessen und der Anwendung verschiedener deskriptiver Entwicklungsmethoden dar. Der Fokus der Richtlinie VDI 2221 ist auf den Maschinenbau zurückzuführen und hat demnach eine gestaltorientierte Sichtweise auf das zu entwickelnde System (MÖHRINGER, 2003). Die prozeduralen und

formalen Aspekte der Richtlinie können beispielsweise für die situationsabhängige Definition von Arbeitsabläufen herangezogen werden. EHRLENSPIEL (2007, S. 317 ff.) merkt zwar an, dass VDI Richtlinie 2221 zwar *die Basis* für die Neuentwicklung von technischen Produkten ist, jedoch durch die strenge, lineare Ablaufdefinition nur bedingt für die Weiterentwicklung technischer Systeme angewandt werden kann.

3.2.2 Entwicklungsmethodik für mechatronische Systeme

Die Komplexität der Entwicklung von Flugzeugsystemen hat zur Entwicklung von Richtlinien und Normen geführt, um die Produkt- und Dokumentationsgüte sowie die Zulassung von Systemen abzusichern. In der Bundesrepublik Deutschland hat sich aus nationalen und internationalen Normierungsbestrebungen das sogenannte V-Modell entwickelt. Das V-Modell ist als Richtlinie für die Planung und Durchführung von (Software-)Entwicklungsprojekten konzipiert. Unter Berücksichtigung des Systemlebenszyklus werden die zu erstellenden Ergebnisse definiert und konkrete Vorgehensweisen zur Ergebniserreichung zur Verfügung gestellt. Darüberhinaus werden den detailierten Arbeitsschritten entsprechende Verantwortlichkeiten der Prozessbeteiligten zugewiesen. Durch den hohen Standardisierungsgrad werden komplexe Projekte systematisiert und damit einfacher umsetzbar. Die Zielerreichung der geforderten Qualität wird abgesichert und die Zertifizierung oder Zulassung des entwickelten Systems erleichtert.

Durch den stetigen Wandel vom rein mechanischen zum mechatronischen Produkt wurde auf Basis des V-Modells die anwendungsorientierte Richtline VDI 2206 geschaffen. Die Richtlinie beschreibt eine domänenübergreifende Entwicklungsmethodik für mechatronische Systeme und stellt dabei die notwendigen Vorgehensweisen, Methoden und Werkzeuge zur Verfügung. Durch die Vorgabe von Prozessbausteinen für wiederkehrende Tätigkeiten werden dem Systementwickler modularisiertes Wissen und konkrete Handlungsanweisungen zur Verfügung gestellt (MÖHRINGER, 2003). Der Fokus liegt auf den frühen Phasen des Entwickelns mit Schwerpunkt Systementwurf.

Das V-Modell der VDI-Richtline 2206 schlägt ein Vorgehen in zwei Abstraktionsebenen nach dem Makrozyklus und dem Mikrozyklus vor (VDI, 2004). Der Makrozyklus stellt das Vorgehensmodell von der Anforderungsdefinition über den Systementwurf und die Systemintegration bis zum marktreifen Produkt dar (Abbildung 3-3). Der Mikrozyklus stellt einen Problemlösungszyklus dar, der innerhalb der einzelnen Entwicklungsphasen angewandt werden kann und basiert auf der systematischen Analyse und Synthese von Teilschritten, die auf diese Weise konkretisiert und iterativ gelöst werden (VDI, 2004).

Zunächst wird im Makrozyklus die Aufgabenstellung konkretisiert und die Produktanforderungen definiert. Die resultierende Systemspezifikation beschreibt konkrete Attribute sowie Zielwerte der Attribute, welche den Maßstab für die spätere Bewertung des Produktes darstellen. Im nachfolgenden *Systementwurf* wird ein Lösungskonzept erstellt und die Produktfunktion in wesentliche Teilfunktionen zerlegt. Die Teilfunktionen werden im *Domänenspezifischen Entwurf* weiter konkretisiert und in Systemelemente durch entsprechende *Modellbildung und Analyse* übersetzt. Das Ergebnis sind erste *Labormuster*, mit denen kontinuierlich die Funktionserfüllung der Teilfunktionen im Gesamtkontext überprüft wird. Die Komponenten eines mechatronischen Systems (Mechanik, Elektronik, Regelungssoftware, Bedienelemente) werden während der Phase *Systemintegration* getestet und in ihrer Funktion optimiert. Weiterhin wird die synergetische Integration der unterschiedlichen Komponenten in verschiedenen Mockups vorangetrieben bis ein *Funktionsmuster* zur Verfügung steht.

Die notwendigen Systemtests bestehen vornehmlich aus Belastungstests, Verträglichkeitsprüfungen, Funktionstests sowie Zuverlässigkeits- und Sicherheitsprüfungen. Die Ergebnisse dienen der Verifikation der Systemspezifikationen und bilden die Basis für die weitere Optimierung der Systemarchitektur und der Komponenten. Nach erfolgreicher Spezifikationsverifikation erfolgt die Zertifizierung und Produktion der finalen Systemarchitektur und die Validierung der Anforderungen durch Qualitätsprüfungen. Der Reifegrad des Systems nimmt mit jeder erreichten Stufe innerhalb des V-Modells zu. Abbildung 3-3 stellt das Vorgehen analog des V-Modells dar und zeigt die Eingliederung des Flugzeugentwurfsprozesses (vgl. Abbildung 2-2) anhand von Meilensteinen. Innerhalb der Prozessschritte des V-Modells treten Iterationen auf, die auf dem Mikrozyklus basieren und in Abbildung 3-3 nicht dargestellt sind.

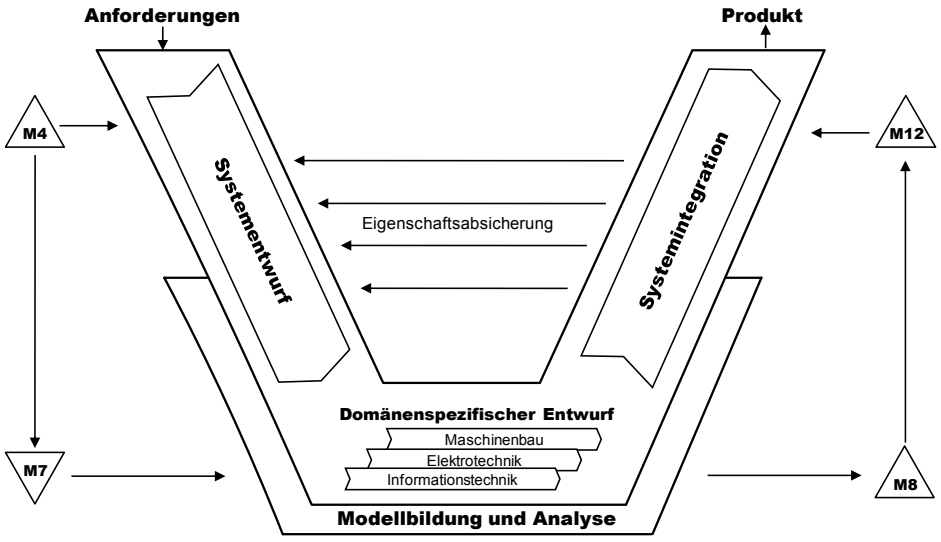


Abbildung 3-3 V-Modell der Entwicklung mechatronischer Produkte
Quelle: VDI (2004); leicht modifiziert

Das Problem bei der Entwicklung von mechatronischen Systemen ist deren Zusammensetzung aus mehreren Subsystemen, die verschiedenen Domänen zugehören. So fordert STAHLKAMP (2009) am Beispiel Integrierter Modularer Avionik (IMA) die Vernetzung von domänenspezifischen Vorgehensmodellen und den Einsatz von Systemarchitekten, welche eine integrierende Rolle übernehmen sollen (Abbildung 3-4). Dabei wird zwar deutlich, dass eine stärkere Vernetzung der Entwicklungsdomänen auf Plattformebene angestrebt wird, wie diese Vernetzung konkret aussehen soll bleibt aber offen.

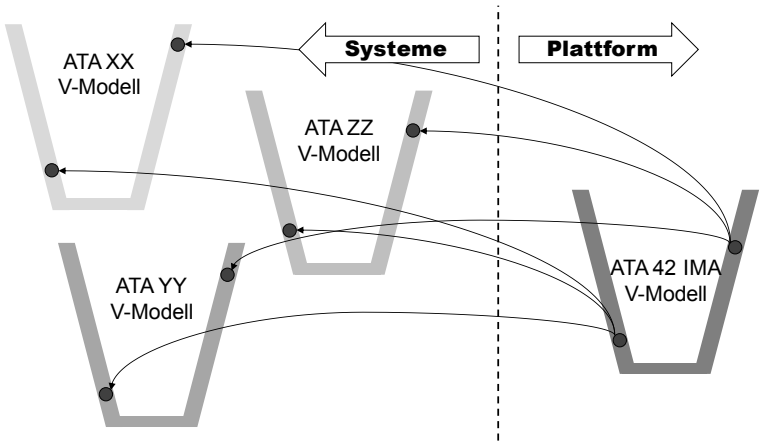


Abbildung 3-4 Integration der Vorgehensmodelle der Systementwicklung
Quelle: STAHLKAMP (2009)

Das V-Modell stellt ein repräsentatives Vorgehensmodell der Systementwicklung dar, da es in unterschiedlichen Branchen und Entwicklungsdomänen Anwendung findet und eine evolutionäre Entwicklung von nationalen und internationalen Entwicklungsstandards darstellt. Aus diesem Grund sollten die genannten Aspekte der Entwicklungsmethodik für mechatronische Systeme nach VDI 2206 bei der Definition von stärker parallelisierten Arbeitsabläufen herangezogen werden. Weiterhin wird mit dem Problemlösungszyklus ein Werkzeug zur Verfügung gestellt, das im Bereich der Verifikation und Validierung domänenübergreifender Fragestellungen die Lösungsfindung beschleunigen kann.

3.2.3 Aerospace Recommended Practices

Die Society of Automotive Engineers (SAE) gibt verschiedene Leitfäden zur Entwicklung von Transportmitteln der Automobil- und Luftfahrtindustrie heraus. Die sogenannten Aerospace Recommended Practices (ARP) stellen eine Sammlung von Wissen über die Entwicklung und Zulassung verschiedener Flugzeugsysteme dar. Beispielhaft wird in der SAE ARP4754A (*Certification Considerations for Highly-Integrated Or Complex Aircraft Systems*) der Entwicklungsprozess für Flugzeugsys-

teme mit Fokus auf Avioniksysteme dargestellt. SAE ARP4761 (*Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*) beschreibt relevante Klassen der Ausfallwahrscheinlichkeit von Flugzeugsystemen und akzeptable Methoden, um Systeme hinsichtlich dieser Ausfallwahrscheinlichkeit zu bewerten. Je nach Kritikalität des zu entwickelnden Flugzeugsystems sind die ARP als Richtlinie zur Systembewertung in den Entwicklungsprozess zu integrieren und entsprechende Nachweise der Anwendungsfälle zu erbringen. Die ARP dienen innerhalb der Entwicklung eines Flugzeugsystems eines bestimmten ATA-Kapitels zur schnelleren Definition von Anwendungsfällen bzw. Testfällen des Systems. Dadurch kann die Absicherung der Anforderungserfüllung zu einem früheren Zeitpunkt erfolgen (engl.: front-loading) und folglich die Robustheit der Systeme gesteigert werden.

3.3 Deskriptive Prozessentwicklungsansätze

3.3.1 Concurrent Engineering

Die Rolle von Flugzeugherstellern hat sich vom klassischen *Hersteller* hin zum *Systemintegrator* verschoben. Das liegt zum einen daran, dass die notwendigen Kompetenzen innerhalb spezifischer Fachdisziplinen stark gewachsen sind und demnach nicht mehr in das Kerngeschäft des OEM fallen. Zum anderen wird durch die zunehmend funktional und modular gestaltete Architektur von Systemen die Herstellung und Entwicklung bei Lieferanten attraktiver. Durch entsprechende Schnittstellengestaltung können auch hoch integrierte Systeme entkoppelt und parallel entwickelt werden (Concurrent Engineering (CE), synonym: Simultaneous Engineering).

Dabei versteht man unter der integrierenden Vorgehensweise des Simultaneous Engineering „die zielgerichtete, interdisziplinäre Zusammen- und Parallelarbeit von Produkt-, Produktions-, und Vertriebsentwicklung mit Hilfe eines straffen Projektmanagements, wobei der gesamte Produktlebenslauf betrachtet wird“ (EHRLENSPIEL, 2007, S.202). Die physische und funktionale Unabhängigkeit von Modulen und die Standardisierung von Schnittstellen innerhalb eines Systems ermöglichen die parallele Entwicklung. CE-Aktivitäten finden unternehmensintern und in zunehmendem Maße unternehmensübergreifend in Entwicklungspartnerschaften statt. Hierbei beschreibt der Begriff Concurrent Engineering eine „Methode zur Verkürzung von Entwicklungszeiten mittels paralleler Initiierung der notwendigen Entwicklungsarbeiten. In diesem simultanen Prozess können auch die Lieferanten einbezogen werden, um die Entwicklung zu beschleunigen und zu optimieren“ (WINTER, MOSENA UND ROBERTS, 2010). In

der Automobilindustrie ist die simultane Projektabwicklung zwischen OEM und Zulieferer bereits seit vielen Jahren etabliert (WOLTERS, 1995), wobei die Kommunikation und der Informationsfluss bei der interdisziplinären Zusammenarbeit deutlich durch die Nutzung kompatibler CAD- und CAE-Systeme beschleunigt werden.

CE hat positive Auswirkungen auf den Produktentstehungsprozess. So lässt sich die Entwicklungszeit mit Hilfe der integrierten Produkterstellung im Gegensatz zu einem sequenziellen Ablauf verkürzen, was zu einer schnelleren Markteinführungszeit führt. Ursachen für die Entwicklungszeitersparnis liegen in der Parallelschaltung der unterschiedlichen Arbeitsabläufe sowie in der engen informatorischen und kommunikativen Verknüpfung der Beteiligten (EHRLENSPIEL, 2007, S.205f.). Ein Beispiel für erfolgreich angewandtes Concurrent Engineering in der Luftfahrtindustrie findet sich am Beispiel Airbus. Hier wurden die Muster A330 und A340 parallel entwickelt. Beide Flugzeugmuster besaßen unterschiedliche Anforderungsprofile und Triebwerksanzahlen, konnten jedoch mit den gleichen Tragflächen ausgelegt werden. Auf diese Weise konnten die Entwicklungszeiten und -kosten beider Flugzeuge im Vergleich zu Einzelentwicklungen stark reduziert werden (KRAUS, 2005).

Die Ziele von CE sind Zeitersparnis bei der Entwicklung und kundenindividuellen Herstellung sowie Kostenreduktion bei gleichzeitiger Steigerung der Qualität (EHRLENSPIEL, 2007). Ein wichtiges Element des CE ist die adäquate Werkzeugverwendung (siehe Kapitel 2.4.3). Um die Datenvielfalt der beteiligten Prozesse beherrschen zu können, sind vor allem Datenbanken und PDM-Systeme für erleichterten Informationsaustausch notwendig (VILSMEIER, 2011). Weiterhin ist die Einführung von semi-automatischen und komplett automatisierten Arbeitsabläufen für die Erreichung der CE-Zielvorgaben notwendig (FIGLAR UND MOHR, 2011). Weitere wichtige Elemente des CE sind die Beseitigung von Kreisschlüssen (engl.: feedback loop) zwischen Prozessen und Prozessbeteiligten, die Früherkennung von Fehlerquellen sowie die Einbindung von Kunden und Zulieferern in der Vorentwicklungsphase (LINDEMANN, MAURER UND BRAUN, 2009; MAUL, 2009).

3.3.2 Systems Engineering

Systems Engineering (SE) ist eine Disziplin des Ingenieurwesens, mit der ein interdisziplinärer Prozess geschaffen werden soll, der es erlaubt die Anforderungen von Kunden und Prozessbeteiligten mit höchster Qualität, Zuverlässigkeit und Kosteneffizienz im Rahmen eines vorgegebenen Zeitplanes zu erfüllen (INCOSE, 2006). SE rückt aufgrund der Berücksichtigung von interdisziplinären Schnittstellen und der wachsenden Unterstützung durch Werkzeuge immer mehr in den Fokus der Konzeptentwickler un-

terschiedlicher Branchen. SE versucht mit den zur Verfügung gestellten Methoden das sogenannte *front-loading* des Entwicklungsprozesses zu erreichen. Damit ist die Fähigkeit gemeint, bereits in der Synthese- und Konzeptphase detaillierte Analysen durchführen zu können, welche die Entscheidungsfähigkeit für oder gegen ein Konzept deutlich erhöhen. Das Risiko von Fehlentscheidungen aufgrund von Informationsmangel oder fehlenden Auswirkungsanalysen wird durch SE-Methoden minimiert.

Der SE-Prozess beinhaltet alle Vorgänge, die notwendig sind um Anforderungen an ein System in Funktionen zu übersetzten und daraus eine Struktur abzuleiten. Um eine funktionale und anforderungsgerechte Systemstruktur zu entwerfen, müssen fortlaufend Verfahren der Validierung und Verifikation angewandt werden (Abbildung 3-5). Verifikation bedeutet in diesem Zusammenhang, die Prüfung der korrekten Abbildung von Anforderung zu Funktion sowie von Funktion zu Systemstruktur. Die Validierung stellt die Prüfung der korrekten Funktion der Systemstruktur in Bezug auf die Anforderungen dar.

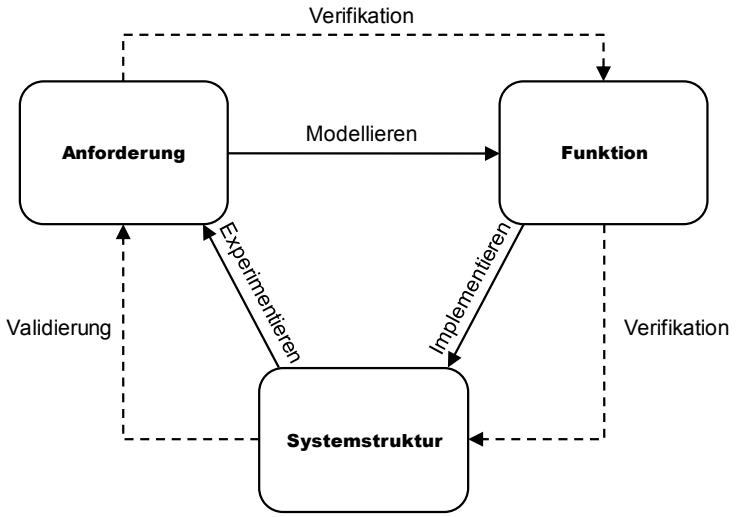


Abbildung 3-5 Verifikation und Validierung im Systems Engineering

Modellbasiertes Systems Engineering (MBSE) propagiert den Wechsel von dokumentenbasierten Verfahren zu rein computergestützten Entwicklungsverfahren. MBSE schafft dabei den formalen Rahmen für die Modellierung von Systemanforderungen und Funktionen, die Ableitung einer physischen Architektur sowie für die Analyse und V&V-Tätigkeiten. Das globale Ziel ist es dabei, die domänenübergreifende Kommunikation zu stärken, Systeme exakter beschreiben zu können und die Integration der Systeme durch die Ermöglichung von Sichten voranzutreiben (FRIEDENTHAL, MOORE, STEINER (2009)).

MBSE erlebt seit einiger Zeit eine hohe Anwendungsbereitschaft in verschiedenen Branchen. Unterstützt wird diese Entwicklung durch die breite Akzeptanz der *Unified*

Modeling Language (UML) als Modellierungssprache. Mit der *Systems Modeling Language* (SysML) steht die Weiterentwicklung von UML als Sprache zur Modellierung komplexer Systeme zur Verfügung, welche ebenso immer weitere Verbreitung findet. SysML unterstützt die Systementwicklung durch die Bereitstellung entsprechender Semantik und Hilfsmitteln wie vordefinierten Diagrammtypen. So werden beispielsweise die Sichten auf ein System durch diese Diagrammtypen explizit vorgegeben.

3.4 Präskriptive Prozessentwicklungsansätze

Dieses Kapitel stellt ausnahmslos sogenannte Standard Engineering Frameworks (auch: Enterprise Architecture Frameworks) aus unterschiedlichen Branchen zusammen. Frameworks sollen dem Management und den Prozessverantwortlichen eines Unternehmens die notwendigen Werkzeuge und Methoden zur Verfügung stellen, um nachhaltige Architekturentscheidungen zu treffen (SCHÖNHERR, 2004).

3.4.1 Modellbasierte Systementwicklung in der Automobilindustrie

Die Automobilindustrie steht wie andere Hochtechnologiebranchen vor der Herausforderung der Integration von Elektrik- und Elektroniksystemen, welche erheblich an der Modelldifferenzierung in Bezug auf Sicherheits-, Funktions- und Marketingargumente beteiligt sind. Verschiedene Ansätze verfolgen das Ziel Methoden und Notationen für die Prozesse des Anforderungsmanagements und Systemdesigns sowie für die Software- und Hardwareentwicklung zu definieren. Grund für die Untersuchung dieser Ansätze waren die generischen, unspezifischen Prozessbeschreibungen herkömmlicher Entwicklungsmethoden und die unzureichende Unterstützung durch Modellierungsmittel. Der Einsatz von UML als Modellierungssprache wurde aufgrund der Vielfalt der UML-Elemente als inadäquat für den spezifischen Einsatz in der Automobilindustrie eingestuft (KLEINOD, 2006).

Vor diesem Hintergrund entstanden unter anderen die Projekte MOSES (*Modellbasierte Systementwicklung in der Automobilindustrie*) und AUTOSAR (*AUTomotive Open System ARchitecture*), welche zur Definition der technischen Architektur von eingebetteten Systemen im Automobil genutzt werden. Beide Projekte weisen charakteristische Elemente auf, die für diese Arbeit signifikant sind. Zunächst erkennen beide Projekte die Relevanz der Möglichkeit zur Darstellung verschiedener Systemsichten an. Die Sichten sollen es ermöglichen, verschiedene Aspekte der logischen und strukturellen Architektur darzustellen. Ein weiteres Charakteristikum ist die Berücksichtigung der Notwendigkeit der Versionierung von erzeugten Modellen. Die

Versionierung sollte zu den prozessbegleitenden Methoden gehören, die im Verlauf des Systementwicklungsprozesses auf die entstehenden Modelle angewendet wird. Dementsprechend wurde innerhalb von MOSES eine Prozesskette für den modellbasierten Entwurf von Systemen in der Automobilindustrie definiert (KLEINOD, 2006). Ein Resultat dieser Erkenntnisse ist die Tatsache, dass die zu entwickelnden Systeme und die in dieser Arbeit diskutierten Systemmodelle nicht von den zugrundeliegenden Prozessen getrennt betrachtet werden dürfen.

3.4.2 Department of Defense Architecture Framework

Das Architekturrahmenwerk des US-Verteidigungsministeriums (kurz: DoDAF) hat das Ziel, eine Systemarchitektur durch eine Reihe von Sichten detailliert zu beschreiben ohne dabei eine Methode vorzugeben, wie diese Sichten erzeugt werden sollen. Einige Derivative des DoDAF stellen das NATO Architecture Framework (NAF) oder das Ministry of Defence (UK) Architecture Framework (MODAF) dar. Allen Frameworks gemein ist die Nutzung einer gemeinsamen Datenbasis, welche durch das Core Architecture Data Model (CADM) beschrieben wird. Aus dieser Datenbasis werden vier vordefinierte Typen von Architektursichten generiert:

- All View (AV)
- Operational View (OV)
- Systems View (SV)
- Technical Standards View (TV)

Die Sicht AV beschreibt dabei den domänenübergreifenden Kontext und Geltungsbereich der Systemarchitektur. Die Sicht OV bereitet Informationen über die operationellen Abläufe wie zum Beispiel dem Informationsaustausch innerhalb eines Systems auf. Durch die Sicht SV werden die Systemelemente und deren Verbindung dargestellt, welche in ihrer Gesamtheit die Systemfunktion sicherstellen. Technische Standards, Regularien und Gestaltungsrichtlinien, welche die Systemarchitektur beeinflussen, werden durch die Sicht TV erfasst und dargestellt.

Innerhalb des DoDAF wird unter einem System die Interaktion eines technischen Elements mit einem oder mehreren Menschen verstanden. Durch die Definition der Sichten, soll ein System von verschiedenen Standpunkten (engl.: view points) aus, verständlich und damit beherrschbar werden. DoDAF fehlt die explizite Vorgabe, wie die Sichten erzeugt werden sollen und welche Sichten aus der vielfältigen Auswahl für die Problemstellung genutzt werden sollen. Die Beantwortung dieser Frage und damit die Methodenauswahl und Vorgehensweise bleibt dem Systemarchitekt überlassen. Durch

diesen Freiheitsgrad lässt sich das Framework in unterschiedlichen Branchen anwenden und wurde mehrfach an spezifische Problemstellungen adaptiert. Dieser Umstand wurde bestärkt durch die Werkzeugunterstützung mittels UPDM (Unified Profile for DoDAF and MODAF), welches durch die Object Management Group (OMG) entworfen wurde, um die UML-Darstellung von DoDAF-Modellen zu vereinheitlichen. Durch ein bereits im Entwicklungswerkzeug enthaltenes Metamodell, welches alle DoDAF Elemente und deren Beziehungen beschreibt, wird die Implementierung erleichtert.

3.4.3 The Open Group Architecture Framework

Im Gegensatz zu DoDAF liegt der Fokus des *The Open Group Architecture Framework* (TOGAF) auf der Methode zur Architekturgenerierung. Hier werden auch nicht die beschreibenden Elemente eines Metamodells zur Verfügung gestellt. Die Architecture Development Method (ADM) bildet den Ansatz zu Entwurf, Planung und Implementierung einer Unternehmensarchitektur. Den Mittelpunkt der ADM bildet das Anforderungsmanagement. Domänenspezifische und bereichsübergreifende Anforderungen sollen besser geplant werden können, um die Zielvorgaben der zu entwickelnden Systeme abzusichern. Weiterhin soll durch TOGAF die Transparenz von Ressourcen verbessert und deren Abhängigkeiten im Unternehmen im entsprechenden Kontext dargestellt werden. Ein weiterer Aspekt, der für diese Arbeit von Relevanz ist, ist das in TOGAF beschriebene Vorgehen zur Implementierung des Frameworks. Die Wichtigkeit der Untersuchung der aktuellen Technologieumgebung eines Unternehmens und die Identifikation von existierender und gewünschter Schnittstellengestaltung zwischen Abteilungen und Werkzeugen werden dargestellt und als essentieller Schritt für nachhaltigen Erfolg beschrieben (THE OPEN GROUP, 2006).

3.5 Erfolgsfaktoren zukünftiger Systementwicklungsmethoden

Dieses Kapitel fasst die Kernaussagen der eben vorgestellten Methoden zusammen und leitet daraus konkrete Leistungsmerkmale für den modellbasierten Entwurf von Flugzeugsystemen ab. Diese Erfolgsfaktoren lassen sich aufteilen in Faktoren der Prozessunterstützung und der Modellierungsunterstützung.

Prozessunterstützung: Entwicklungsmethoden sollten die parallele Bearbeitung von Arbeitsschritten sowie die gleichzeitige Bearbeitung durch mehrere Personen erleichtern und als Folge die Entwicklungszeit verkürzen. Für eine weitreichende Akzeptanz sollten die Prinzipien vorhandener Vorgehensmodelle berücksichtigt und erweitert werden.

Eine integrierende Entwicklungsmethode sollte stets eine beispielhafte Anwendung zur Demonstration der konkreten Implementierbarkeit und Anforderungserfüllung aufweisen. Das Fehlen einer Richtlinie zur erfolgreichen Implementierung erschwert die praktische Anwendung vieler Methoden. So beschreibt LANGERMANN, dass dem Vorgehensmodell nach VDI ein konkreter Vorschlag zur Umsetzung der Vernetzung verschiedener Systeme in der Entwicklung fehlt (LANGERMANN, 2009, S.9).

Besonderes Augenmerk sollte auf Methoden liegen, die ihren Schwerpunkt in der Konzeptphase besitzen, da hier das größte Wertschöpfungspotenzial liegt. In der Konzeptphase werden durch die Untersuchung einer Vielzahl von Konzeptalternativen potenziell robustere Produkte definiert, welche Wettbewerbsvorteile kreieren können. Um dieses *front-loading* des Entwicklungsprozesses zu erreichen, sind Methoden zur Unterstützung der Vorentwurfsphase (vgl. Kapitel 2.3.1) sowie die Standardisierung von Schnittstellen und Austauschformaten notwendig. Dazu zählt die Vermeidung von manuellem Informationsaustausch zwischen Domänen wie in Abbildung 3-6 dargestellt. Nur so lassen sich redundante Datenbestände reduzieren und domänenübergreifende Tätigkeiten automatisieren.

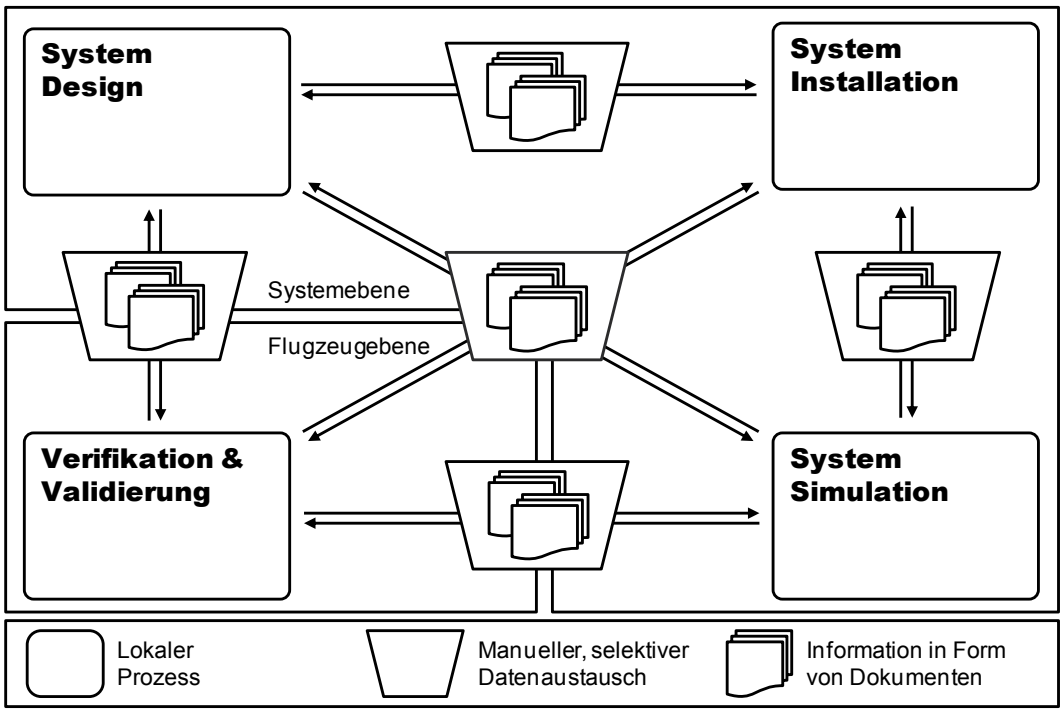


Abbildung 3-6 Manueller Informationsaustausch zwischen Domänen

Die Auswahl der Softwarewerkzeuge kann den automatisierten Austausch von Informationen zwischen Domänen behindern und sollte deshalb bei der Gestaltung zukünftiger Unternehmensarchitekturen besondere Beachtung finden. Verschiedene Ansätze aus dem Bereich der Geschäftsprozessmodellierung oder der sogenannten Enterprise Architecture Integration (EAI) bieten Ansätze zur Lösung der genannten Problematik,

fokussieren jedoch stets nur auf die Integration spezifischer Teilbereiche (GÜNZLER, 2005). Die Verwendung von modellbasierten Verfahren, die unter Nutzung eines zentralen Systemmodells (oft als Produktmodell referenziert) domänenübergreifende Tätigkeiten ermöglichen, wurde als ganzheitlicher Ansatz identifiziert (u.a. BROY ET AL., 2009). Abbildung 3-7 zeigt schematisch den modellbasierten, automatischen Datenaustausch zwischen Entwurfsdomänen unterschiedlicher Hierarchie.

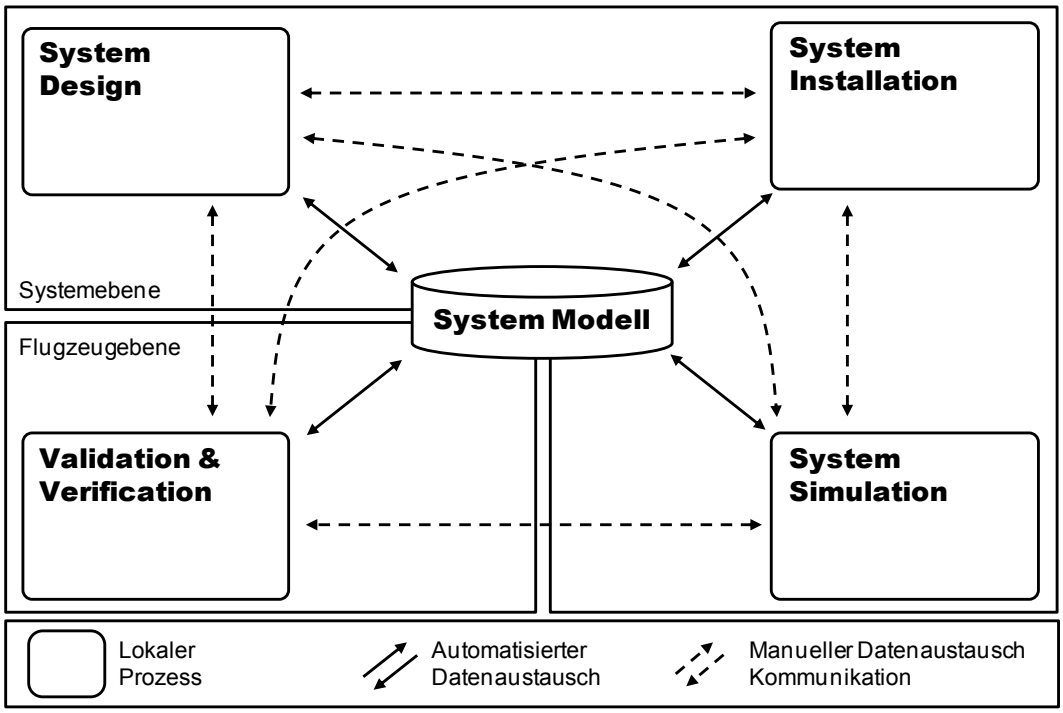


Abbildung 3-7 Modellbasierter Datenaustausch

Modellierungsunterstützung: Das Modell eines Systems sollte alle Sichten auf das System integrieren und damit die Basis für multi-disziplinären Entwurf bilden. Dabei dient das Systemmodell als zentrales Element zur Konsolidierung von Daten in einem Entwicklungsprozess. Ein Modell sollte die Möglichkeit bieten, Anforderungen, Funktionen und Attribute eines Systems zu erfassen und Relationen zwischen Systemelementen abzubilden. Durch die Definition von anwendungsspezifischen Datensätzen innerhalb des Systemmodells werden lediglich benötigte Systeminformationen ausgetauscht und damit explizit auf überladene Datensätze verzichtet, welche durch die Verwendung von herkömmlichen Datenaustauschformaten entstehen. Konkret bedeutet dies die projektabhängige Neudefinition einer Systembeschreibung ohne dabei auf fest definierte Datenstrukturen eines Dateiformates wie STEP zurückgreifen zu müssen. Der erstmalige Modellierungsaufwand eines Systemmodells kann je nach Projektumfang erheblich sein.

Weiterhin wurde dargestellt, dass die Entwicklung eines Systems ein hochgradig interaktiver Ablauf zwischen verschiedenen Fachgebieten ist. Der Modellierungsvorgang

darf deshalb nicht nur den reinen Austausch von Attributen zwischen Werkzeugen fokussieren, sondern muss die umgebenden Prozesse berücksichtigen und abbilden können. Der Modellierungsaufwand wird durch diese klare Prozessdefinition und verbesserte Prozessabläufe wieder relativiert.

Ein Systemmodell sollte sich zudem einer allgemeinen, offenen Sprache zur Modellrepräsentation bedienen, welche die Schnittstellengestaltung zu der im Entwicklungsprozess verwendeten Software ermöglicht und damit die Interoperabilität zwischen Domänen herstellt.

Vor diesem Hintergrund und den Defiziten der aktuellen Systementwicklungsprozesse innerhalb der aktuellen Entwicklung von Flugzeugsystemen (vgl. Kapitel 2.4) sollten die in Tabelle 3-2 dargestellten Faktoren für die erfolgreiche, modellbasierte Systementwicklung berücksichtigt werden. Ein entsprechender Ansatz für ein Modell zur Beschreibung von Flugzeugsystemen wird im folgenden Kapitel hergeleitet.

Tabelle 3-2 Erfolgsfaktoren der Systemmodellierung

Fokus	#	Erfolgsfaktoren neuer Methoden der Systementwicklung
Prozess	1	Parallele Bearbeitung von Arbeitsschritten möglich
	2	Gleichzeitige Bearbeitung eines Modells durch mehrere Personen möglich
	3	Potential zur Automatisierung von bisher manuellen Tätigkeiten
	4	Integrationsmöglichkeit in vorhandene Vorgehensmodelle, IT- Landschaft
	5	Unterstützung der Konzept- und Synthesephase
	6	Vermeidung von redundanten Daten durch geeignete Prozess- und Datenmodelle
	7	Erleichterung der Implementierbarkeit durch detailliertes Anwendungsbeispiel
Modellierung	8	Verwendung eines zentralen Systemmodells (Produktmodells)
	9	Modellierbarkeit von anwendungsspezifischen Datensätzen
	10	Ermöglichung von Systemsichten
	11	Abbildung von Anforderungen, Funktionen und Attributen
	12	Abbildung von Systemelementen und Relationen
	13	Erfassung von Prozessen und Systemen sowie deren Abhängigkeiten
	14	Verwendung offener Sprachstandards zur Modellierung des Systemmodells

4 SITUATIVE UNTERSTÜTZUNG DER ENTWICKLUNG VON FLUGZEUGSYSTEMEN

Innerhalb der vorangegangenen Kapitel wurde beschrieben, dass der Entwurf von Systemen aus dem Umfeld der Luftfahrt durch die verbesserte Nutzung von Informationstechnologie eine höhere Effizienz erreichen kann. Davon ist insbesondere die tiefere Integration von Prozessen, Modellen und verwendeter Software betroffen, welche durch verschiedenste Forschungsansätze und entsprechende Theorien zu erreichen versucht wird. Für die situative Unterstützung der Entwicklung von Flugzeugsystemen sollen zunächst die Grundlagen der Modellbildung sowie verschiedene Modellierungsansätze diskutiert werden. Auf Basis der Diskussion und vor dem Hintergrund dieser Arbeit wird die Verwendung eines zentralen Systemmodells weiterverfolgt, das für den Austausch von produktdefinierenden Daten an zentraler Stelle genutzt werden kann und in seiner Struktur nachfolgend hergeleitet werden soll.

4.1 Begriffliche Abgrenzung

Der Begriff *Modell* beschreibt in den unterschiedlichsten Bereichen der Wissenschaft und Technik die Nachbildung eines realen oder gedanklichen Vorbildes, wobei die vereinfachende Erfassung der zugrundeliegenden Struktur und Funktion die Untersuchung des Vorbildes erleichtert oder ermöglicht (vgl. BUNGARTZ ET AL., 2009). Je nach Anwendungsfall des Modells existieren verschiedene Modellarten, u.a. Strukturmodelle, Verhaltensmodelle oder Vorgehensmodelle, die sich je nach Branche oder Einsatzszenario in ihrer Ausprägung unterscheiden können.

Trotz der unterschiedlichen Einsatzszenarien existieren grundsätzliche Gemeinsamkeiten in der Aufgabe und Bildung von Modellen. Zunächst erlauben alle Modelle die Erfassung der Struktur und Funktion eines Systems mit dem Ziel, das System unter verschiedenen Aspekten zu analysieren, was mit dem realen Vorbild nur unter großen Kosten, Risiken oder Zeitaufwand möglich wäre (vgl. SCHILLER, 2009). Eine weitere Gemeinsamkeit besteht in der Modellbildung. Hier werden aus der Gesamtheit aller

Eigenschaften des realen Vorbildes nur gewisse, für die aktuelle Problemstellung relevante Aspekte extrahiert, wohingegen der unmaßgebliche Rest vernachlässigt wird. Aus der daraus resultierenden Vereinfachung und Abstraktion besitzen alle Modelle nur einen eingeschränkten Gültigkeitsbereich gemäß der Zielsetzung der Modellbildung. Werden verschiedene Anwendungsfälle betrachtet, kann es von Vorteil sein, mehrere Modelle zu erstellen, um die Handhabbarkeit der Modelle nicht zu kompromittieren.

Nach der begrifflichen Abgrenzung des Modells lässt sich der Begriff des *Produktmodells* bzw. des *Systemmodells* besser einordnen. Nach ANDERL (2011) ist ein Produktmodell „die formale Beschreibung aller Informationen zu einem Produkt über alle seine Phasen des Lebenszyklus hinweg“ und als Resultat des Produktentwicklungsprozesses zu verstehen. DYLA (2002) beschreibt in Anlehnung an WASMER (1998) ein Produktmodell als Informationsmodell, „welches alle im jeweiligen Kontext relevanten Informationen eines Produkts abbildet. Es ist aus formal beschriebenen Modellschemata, inklusive Definitionen der beschriebenen Sachverhalte, aufgebaut. Dadurch werden Datenstrukturen festgelegt, mit denen individuelle Produkte im Rechner abgebildet werden können“. Analog dazu betrachtet das Projekt ARTWORK (2003) ein Produktmodell „als Gesamtheit aller relevanten Produktinformationen“, welches zur Abbildung von Lebenszyklusinformation eines Systems in einem domänenübergreifenden Kontext dient. In diesem Zusammenhang wird auch von einem integrierten Produktmodell (oder Systemmodell) gesprochen (ARTWORK, 2003; GÜNZLER, 2005). Diese zuletzt genannte Definition soll auch für den Arbeitsumfang dieser Arbeit gelten.

4.2 Modellierungsansätze

Grundsätzlich stellt die Modellbildung einen Abstraktionsprozess dar, der in verschiedenen Branchen eingesetzt wird, um die Komplexität eines realen oder abstrakten, gedanklichen Systems auf ein bewältigbares Maß zu reduzieren (vgl. z.B.: BRANDSTÄTTER, 2009; OMG, 2011). Das Modell soll dabei die relevanten Eigenschaften und Funktionen des realen oder abstrakten Systems weiterhin ausreichend ausdrucksstark wiedergeben können. Der Modellbildungsprozess beschreibt die Transformation, die ein bestimmtes System, wie zum Beispiel ein Avioniksystem, in ein abstraktes Modell überführt. Der generische Abstraktionsprozess der Modellbildung ist in Abbildung 4-1 als Zyklus beschrieben, der iterativ ausgeführt wird, bis das Modell die Realität so wiedergibt, wie für die Anwendung des Modells notwendig.

Während des iterativen Modellbildungsprozesses werden die grundlegende Struktur, die Funktionalität und das Verhalten eines realen (oder gedanklichen) Systems abstrahiert. Das so entstehende abstrakte und rein formale Modell muss in der Folge in ein oder mehrere Simulationsmodelle (bzw. Berechnungsmodelle) implementiert werden. Auf Basis des Simulationsmodelles können die gewünschten Experimente mittels der Erzeugung von Modell- und/ oder Datenvarianten durchgeführt werden. Die Ergebnisse der Experimente müssen schließlich gegenüber dem realen Systemverhalten validiert werden. Sind Abweichungen von der (erwarteten) Realität festzustellen, müssen die Simulationsannahmen überprüft werden und sowohl das abstrakte Modell als auch das Simulationsmodell verifiziert werden.

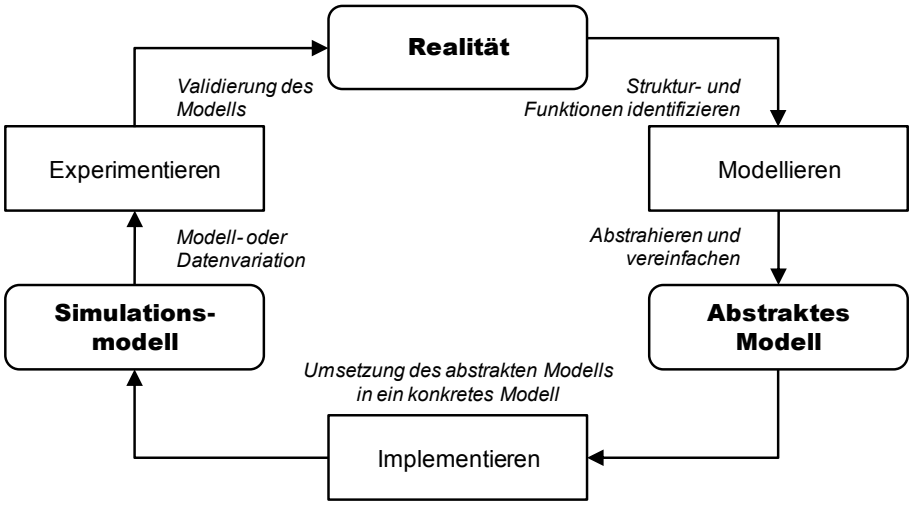


Abbildung 4-1 Abbildung der Realität durch Modelle
Quelle: BRANDSTÄTTER (2009)

Aus Abbildung 4-1 ist zu entnehmen, dass die Vorgänge des *Modellierens*, *Implementierens* und *Experimentierens* essentielle Bestandteile der durchgängigen Modellbildung sind. Je nach Anwendungsfall oder Branche haben sich für jede dieser Tätigkeiten spezielle Handlungsmuster etabliert.

NEGELE (1998) stellt eine Vorgehensweise zur Modellierung von technischen Systemen vor (vgl. Abbildung 4-2). Zunächst wird das System klar von seiner Umwelt unterschieden und die Systemgrenze festgelegt. Daraufhin sind alle Systemelemente verschiedener Abstraktionsebenen festzulegen und diese Elemente durch Relationen zu verknüpfen. Durch die Definition des Relationstyps können Ein- und Ausgangsgrößen exakt spezifiziert werden und damit die strukturelle, deskriptive Modellierung abgeschlossen werden. In einem zweiten Modellierungsschritt wird durch die Definition der Elementparameter sowie der qualitativen und quantitativen Systemfunktionen das Modell um funktionale Aspekte erweitert. Das Vorgehen kann iterativ erfolgen, bis der gewünschte Detailgrad des Systemmodells erreicht ist.

Wie eben beschrieben, fokussieren theoretische Ansätze zur Modellbildung meist ein *Top-Down-Vorgehen* (bzw. *vom Groben zum Feinen* oder *vom Abstrakten zum Detail*). Diese Tatsache steht allerdings im Gegensatz zur praktischen Implementierung der Theorie mittels der am Markt vorhandenen Werkzeuge, wo Systeme sukzessive aus Einzelelementen aufgebaut werden (*Bottom-Up-Vorgehen*).

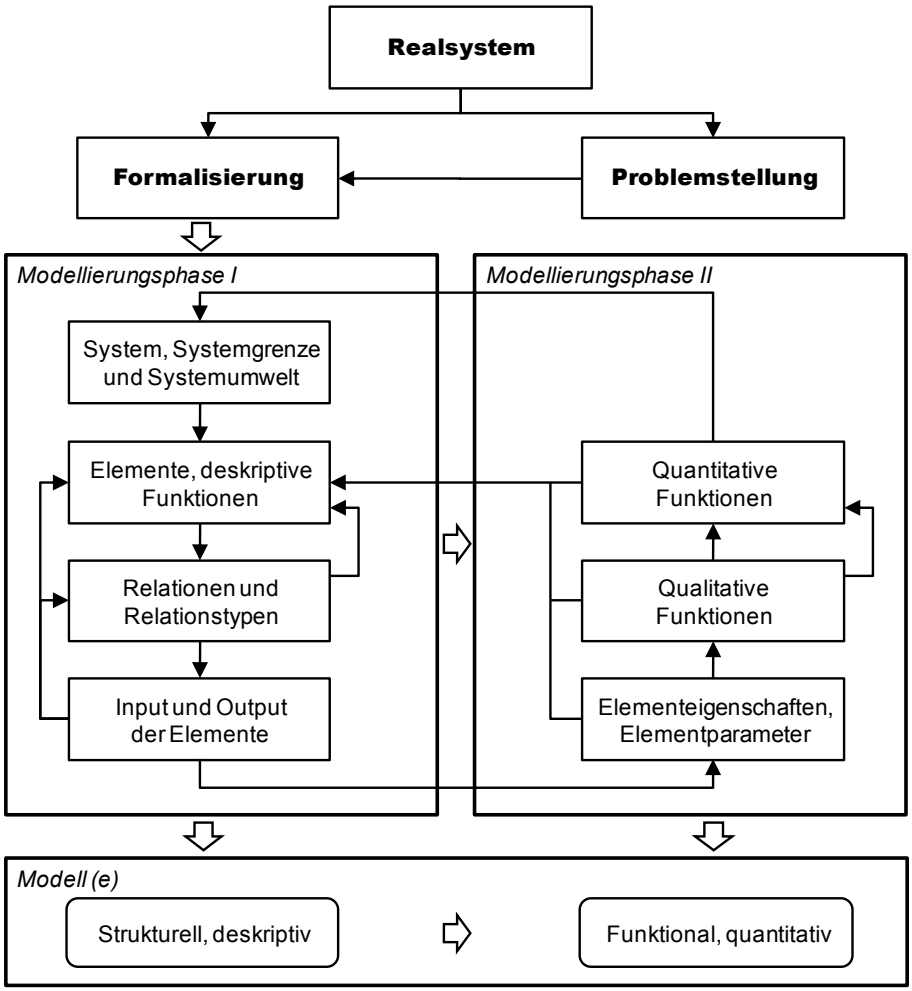


Abbildung 4-2 Vorgehensweise der Modellbildung technischer Systeme
Quelle: Negele (1998); stark modifiziert

Nachdem für die Erstellung von Modellen in der Luftfahrt meist mehrere Teilmodelle verschiedener Disziplinen erstellt werden (vgl. Kapitel 3.5), haben sich mehrere Ansätze für die tatsächliche Generierung von Modellen etabliert. Diese Ansätze unterscheiden sich vornehmlich durch die *Kompatibilität* der Partialmodelle.

Unter Kompatibilität kann im Sinne von Verträglichkeit, angewendet auf technische Systeme, verstanden werden, dass zwei (autonome) Systeme/Partialsysteme zueinander kompatibel sind, wenn sie unter spezifischen Bedingungen miteinander interagieren können bzw. die geforderte Funktionalität und das geforderte Verhalten besitzen. Schnittstellenkompatibilität zwischen Systemen (auch: Teilsystemen/ Komponenten) ist vorhanden, wenn an einer gemeinsamen Schnittstelle statische und dynamische

Kompatibilität herrscht. Dabei ist es unerheblich, ob es sich um eine Elektrotechnik-, Mechanik- oder Softwareschnittstelle handelt; die Schnittstellenkompatibilität ist also domänenunabhängig. Modelliert man zwei Subsysteme, die an einer Schnittstelle eine unidirektionale Relation besitzen, und stimmt das gesendete Signal des Senders mit dem vom Empfänger erwarteten Signal sowohl statisch (unveränderliche Eigenschaften) als auch dynamisch (veränderliche Eigenschaften) überein, spricht man von Schnittstellenkompatibilität (BRANDSTÄTTER, 2009).

Aufgrund der Diskrepanz zwischen theoretischen und praktischen Modellierungsansätzen sollen in den folgenden Abschnitten, die gängigen Modellierungstechniken in der industriellen Praxis vorgestellt und untersucht werden. Dabei werden die Modellierungstechniken durch die Kompatibilität ihrer Partialmodelle unterschieden.

4.2.1 Entwicklungsplattformen

Unter dem Schlagwort „einheitliche Entwicklungsplattform“ (engl.: single system standardization) werden Ansätze verstanden, die den unternehmensweiten Einsatz eines standardisierten Softwarepaketes zur digitalen Modellerstellung propagieren. Diese Entwicklungsplattformen halten für spezielle Entwicklungstätigkeiten eigene Module bereit, die je nach Anwendungsfall auf die Entwicklungsplattform aufgesetzt werden können. Der große Vorteil einer solchen Entwicklungsumgebung ist die garantierte Interoperabilität der erstellten Modelle, die Verwendung der gleichen Softwareversion vorausgesetzt. Allerdings sind mit dem Einsatz einer einheitlichen Entwicklungsplattform bei OEM und allen Zulieferern große finanzielle Hürden und Abhängigkeiten von einem Softwarehersteller verbunden. Weiterhin ist die adäquate Unterstützung der angebotenen Module für domänenspezifische Tätigkeiten nicht in jedem Fall zu erwarten. Gerade im Bereich der Entwicklung von luftfahrttechnischen Erzeugnissen ist durch die Komplexität der Einzeltätigkeiten eine breite Landschaft an Werkzeugen entstanden, deren Ersatz durch kommerzielle Anwendungen, die einen breiten Markt bedienen wollen, unwahrscheinlich erscheint.

Vor dem Hintergrund der identifizierten Erfolgsfaktoren zukünftiger Systementwicklung (vgl. Kapitel 3.5) wird in vielen Bereichen versucht, das Systemverhalten von komplexen Systemen bereits in sehr frühen Stadien des Entwurfszyklus zu antizipieren. Zu diesem Zweck werden vermehrt Simulationsmodelle innerhalb einer autonomen, von den Fachdomänen entkoppelten Entwicklungsumgebung erstellt (vgl. LISCOUET-HANKE, 2008; MAUL, 2009). Diese Art der modellbasierten Systementwicklung eignet sich für Parameterstudien und stellt damit die Basis für abteilungsübergreifende Diskussionen sowie von generellem Systemverständnis dar. Die integra-

tive Darstellung eines Systemmodells unter Berücksichtigung aller Domänen im Entwurfsprozess ist mit diesen Systems-Engineering-Modellen jedoch nicht möglich.

4.2.2 Interpretation von Datenformaten

Wie bereits in Kapitel 2.4.3 angedeutet worden ist, besteht die Möglichkeit Daten zwischen verschiedenen Werkzeugen durch die Interpretation bzw. Übersetzung von Modellen auszutauschen. Dabei besteht zum einen die Möglichkeit der Verwendung von maßgeschneiderten Übersetzern zwischen zwei Dateiformaten und zum anderen der Einsatz von neutralen Datenformaten, welche sowohl vom Quell- als auch vom Zielsystem interpretiert werden können. Der Einsatz von Interpretationsalgorithmen stellt sich in der Praxis als sehr aufwändig dar, da jeder Softwarewechsel bzw. jede neue Softwareversion erheblichen Aufwand in der Neuerstellung bzw. Anpassung der Schnittstellendefinition mit sich bringt. Maßgeschneiderte Übersetzer ermöglichen den maximal möglichen Informationsaustausch zwischen zwei Modellen und sind lediglich durch die Interpretationsfähigkeit des Zielsystems beschränkt.

Neutrale Datenformate haben hingegen per Definition ein limitiertes Vermögen Modelldaten abzubilden und bieten demnach nur in bestimmten Anwendungsbereichen sinnhafte Verwendungsmöglichkeit. Eine weitere Einschränkung der Übersetzung von Modelldaten in neutrale Datenformate ist der aktuell stark ausgeprägte Fokus auf Geometriedaten.

Nicht zu vergessen ist letztlich der manuelle Austausch spezifischer Systemdaten zwischen verschiedenen Modellen, der in kleinen Entwicklungsteams durchaus Berechtigung finden kann.

4.2.3 Systembeschreibung auf Metaebene

Aufgrund der genannten Defizite der beiden eben genannten Ansätze, kann der Wunsch geäußert werden, ein zentrales Metamodell zu definieren, das relevante Lebenszyklusinformationen aus vorhandenen Partialmodellen an zentraler Stelle erfasst und verschiedene Sichten auf die Systeminformationen ermöglicht. Dieser Ansatz basiert auf der Definition eines formalen Metamodells, welches die generische Beschreibung eines Systems ermöglicht und damit die freie Verwendung domänenspezifischer Entwicklungswerkzeuge zulässt.

JECKLE (2000) definiert ein Metamodell als „ein Modell, das ein Modell beschreibt“. Der Begriff Metamodell ist aus dem griechischen Präfix *Meta* abgeleitet, welches im Zusammenhang dieser Arbeit mit „über“ übersetzt werden kann. Ein Metamodell stellt

also die formale Beschreibung eines Modells dar und muss hierfür über geeignete Sprachelemente verfügen. Notwendige Bedingung an das Metamodell ist die Fähigkeit alle während der Entwicklung entstehenden, produktdefinierenden Daten in dem Modell ablegen zu können. Produktdefinierende Daten sind laut SPUR UND KRAUSE (1997) alle einem definierten Produkt zugeordnete Daten wie Geometrie, Topologie, technologische Information und organisatorische Daten. Der verbreitete Einsatz von reinen CAD-Systemen hat dabei zu einer Fokussierung auf den reinen Geometriedatensatz geführt, technologische und organisatorische Informationen stehen für eine weitere Anwendung nicht mehr zur Verfügung oder können erst gar nicht erfasst werden. Im Rahmen einer Anwendung von Systemmodellen ist deshalb die gesamte Bandbreite der im Entwurf eines Systems eingesetzten Softwarewerkzeuge zu betrachten und entsprechende Schnittstellen vorzuhalten, um relevante, produktdefinierende Daten austauschen zu können.

Nachdem für die theoretischen Grundlagen sowie die praktische Umsetzung eines Metamodells bisher keine einheitliche Richtung vorgegeben wird, sollen im folgenden Kapitel die Anforderungen an Modelle im Allgemeinen diskutiert werden, um anschließend ein formales Metamodell für die Beschreibung von Flugzeugsystemen ableiten zu können.

4.3 Anforderungen an Modelle

Es existieren bereits verschiedene, generische Modelle zur Beschreibung von Produkten, die aber durch ihren Anspruch auf Allgemeingültigkeit enorme Ausmaße in Bezug auf enthaltene Modellierungskonstrukte und Syntax erreicht haben. Ein Beispiel ist das Core Product Model (CPM) des National Institute of Standards and Technology (NIST), was aber als Grundlage für die weiteren Betrachtung eines Produktmodells dient (CPM: vgl. FENVES ET AL., 2005). Ergänzend wird das Produktmodell nach GÜNZLER (2005) für die Adaption an die Problemstellung dieser Arbeit herangezogen, wobei zunächst die Anforderungen an Metamodelle diskutiert werden. GÜNZLER (2005) und BRANDSTÄTTER (2009) führen folgende Anforderungen an Metamodelle an:

Systemsichten: Wie bereits in Kapitel 2.3.3 beschrieben worden ist, unterscheiden sich die Sichten auf ein System je nach der Rolle des Entwicklers innerhalb des Entwicklungsprozesses erheblich. Ein Metamodell muss deshalb die Möglichkeit bieten, individuelle Sichten auf ein Produkt zu ermöglichen, um im aktuellen Kontext relevante Systemeigenschaften darstellen zu können, ohne das Gesamtmodell zu kompromittieren.

Effizienz und Ausdruckskraft: Um ein Metamodell effizient erstellen zu können, muss den Anwendern der Methode eine angemessene und überschaubare Anzahl an Elementen zur Verfügung stehen, mit Hilfe derer eine Systemstruktur zu erfassen ist. Um eine entsprechende Akzeptanz der Methode zu erreichen, ist für die Metamodellierung auf moderne Ansätze wie beispielsweise die UML zurückzugreifen (vgl. Kapitel 5.2). Gleichzeitig muss durch die zur Verfügung stehenden Elemente des Metamodells eine möglichst hohe Ausdruckskraft erreicht und alle systemrelevanten Aspekte abgebildet werden. Dazu gehören zum einen die Elemente eines Systems und deren geometrische und physikalische Attribute und zum anderen die Beziehungen zwischen den Einzelelementen.

Durchgängigkeit: Ein integrierendes Systemmodell muss Systemdaten aus allen Phasen des Lebenszyklus beschreiben können. Deshalb ist bereits zu Beginn einer Entwicklung ein Metamodell zu entwerfen, dass diesen Umstand berücksichtigt. Natürlich können im Laufe der Entwicklung Modelle zu Modellen höheren Detaillierungsgrades migriert werden, die Sprachelemente der Metaebene sollten aber ausreichend ausdrucksstark sein, alle Aspekte eines Systems abzubilden.

Unabhängigkeit und Kompatibilität: Wie in Kapitel 2.4.3 beschrieben, konzentrieren sich heute viele Prozesse an den zur Verfügung stehenden Werkzeugen, die meist proprietäre Datenformate erzeugen und damit eine Vielzahl von inkompatiblen Partialmodellen bedingen. Für die Erstellung von Metamodellen ist deshalb eine werkzeugunabhängige Modellierungstechnik anzuwenden, um Änderungen an den Metamodellen, die durch die Wechsel oder Weiterentwicklung des Werkzeugs entstehen, zu vermeiden. Die erstellten Metamodelle sollten kompatibel zu allgemeinen Formaten wie UML oder STEP sein und zudem die Möglichkeiten bieten, über Schnittstellen mit proprietären Datenformaten zu kommunizieren. Systemmodelle sollten folglich unabhängig von proprietären Formaten und Software erzeugt werden, um die gewünschte Integrationsgrundlage für verschiedenste, domänenspezifische Partialmodelle zu bilden.

Umsetzbarkeit und Handhabbarkeit: Um mit Hilfe der Metamodellierung die erfolgreiche Integration unterschiedlichster Partialmodelle innerhalb eines Unternehmens umzusetzen, sollte die Metamodellierung auf zeitgemäßen Technologien wie beispielsweise der Objektorientierung basieren. Um die Umsetzbarkeit weiter zu verbessern, sollte die gewählte Technologie möglichst von frei zugänglichen Werkzeugen unterstützt werden, deren Funktionsumfang auf die Problemstellung angepasst werden kann. Weiterhin sollte die Handhabbarkeit des Systemmodells gewährleistet sein, um die regelmäßigen Tätigkeiten der Modellerstellung und –änderung ohne Schwierigkei-

ten ausführen zu können. So kann zum einen ein überschaubares und verständliches Metamodell gefordert werden, zum anderen eine übergreifend akzeptierte Modellierungstechnologie. Für die Umsetzbarkeit und Handhabbarkeit ist weiterhin ein entsprechendes Werkzeug notwendig, das den Entwickler aktiv unterstützt und unabhängig von der vorhandenen IT-Infrastruktur eingesetzt werden kann.

4.4 Metamodell Flugzeugsystem

Vor dem Hintergrund der beschriebenen Anforderungen gilt es nun, ein Metamodell für den speziellen Einsatz der Modellierung von Flugzeugsystemen zu definieren. Hierzu werden die Aspekte *Segmentierung*, *Strukturierung*, *Typisierung* und *Parametrisierung* diskutiert in Anlehnung an ARTWORK (2003), SASAKI (2004), GÜNZLER (2005) und KATZKE (2008).

4.4.1 Segmentierung

Zunächst stellt sich die Frage, wie die unterschiedlichen Sichten der am Systementwurf beteiligten Domänen realisiert werden sollen. Als Sicht wird die anwendungsspezifische Darstellung eines Ausschnitts an Systeminformationen bezeichnet, die innerhalb einer Domäne benötigt oder erzeugt wird. Dafür stehen generell zwei Ansätze zur Verfügung – die enge Kopplung und die lose Kopplung von Modellen (engl.: tightly coupled & loosely coupled). Unter enger Kopplung versteht man die Verwendung eines von allen Domänen gemeinsam benutzten Modells, wodurch Sichten darauf beschränkt sind, die Informationen des Modells unterschiedlich darzustellen. Im Gegensatz dazu, besteht innerhalb der losen Kopplung ein Metamodell aus mehreren Teilmodellen, die untereinander über Querbezüge verbunden sind. Die Wahl des Segmentierungsansatzes hängt vom Anwendungsfall ab und soll nun diskutiert werden.

Innerhalb der engen Kopplung existiert lediglich ein zentrales Metamodell, welches die globale Systemstruktur darstellt und deshalb als Kommunikationsmodell innerhalb des Entwicklungsprozesses benutzt werden kann (Abbildung 4-3). Eine zentrale Datenquelle vermeidet Redundanz und sichert damit implizit die Konsistenz der unternehmensweit verwendeten Daten. Die Schnittstellengestaltung zwischen einem Metamodell und den domänenspezifischen Partialmodellen ist weniger aufwändig als zwischen lose gekoppelten Modellen. Allerdings benötigt der eng gekoppelte Ansatz einen erhöhten Abstimmungsbedarf zwischen den Entwicklungsdomänen, da nur ein universell gültiges Modell mit einer definierten Auswahl von Eigenschaften genutzt wird. Je komplexer das zu entwickelnde System wird, desto umfangreicher wird der domänenübergreifende Datenaustauschbedarf und damit das Metamodell. Gerade

wenn graphenorientierte Modellierungsverfahren eingesetzt werden, können diese Ansätze aufgrund der Vielzahl abzubildender Elemente an die Grenzen der verständlichen Darstellung gelangen.

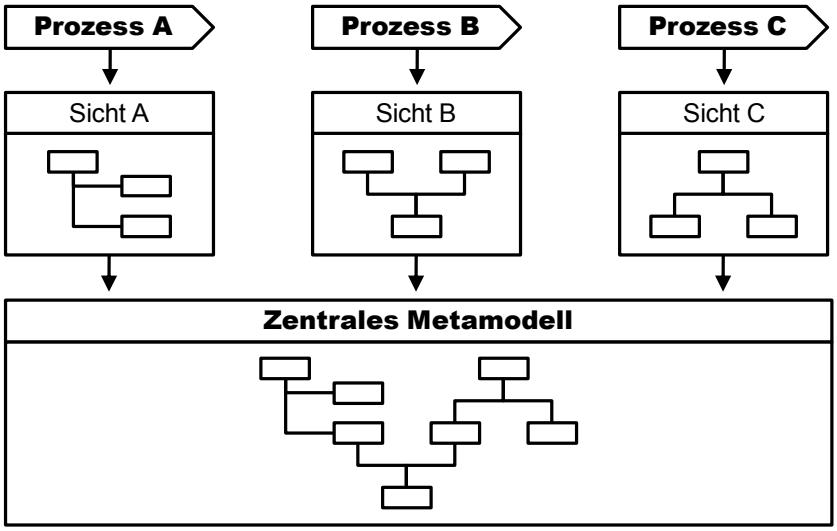


Abbildung 4-3 Ansatz der engen Kopplung
Quelle: GÜNZLER (2005, S.76), leicht modifiziert

Die lose Kopplung hingegen fasst Sichten als eigenständige Modelle auf und verknüpft diese über Konsistenzbedingungen. Der Vorteil von mehreren lose gekoppelten Metamodellen ist die verbesserte Handhabbarkeit in der Erstellung und Modellierung der abzubildenden Systeme. Ein weiterer Aspekt ist die Modifizierbarkeit einzelner Modelle innerhalb des evolutionären Entwicklungsablaufs. Jedes System erreicht im Laufe seiner Entwicklung verschiedene Stufen, die sich durch ihren unterschiedlichen Detaillierungsgrad unterscheiden. Da nicht zu Beginn einer Entwicklung alle zu modellierenden Aspekte bekannt sind, muss die Möglichkeit bereitgehalten zu werden, die Metamodelle analog der Entwicklung der domänenspezifischen Modelle anzupassen. Diese Weiterentwicklung von Modellen auf der Metaebene kann innerhalb des Ansatzes der losen Kopplung unabhängig von anderen Partialmodellen erfolgen. Je nach Art und Ausführungshäufigkeit der Konsistenzprüfungen zwischen den Metamodellen, können innerhalb der losen Kopplung auch inkonsistente Modelle auftreten, welche die Untersuchung von alternativen Architekturen zulässt. Der Ansatz der losen Kopplung ist in Abbildung 4-4 beispielhaft dargestellt.

Zusammenfassend lässt sich feststellen, dass die Wahl des Segmentierungsansatzes von der industriellen Problemstellung abhängt. Die enge Kopplung und Verwendung eines einzelnen Metamodells eignet sich vor allem für Szenarien mit einem begrenzten Koordinationsbedarfs zwischen verschiedenen Domänen wie beispielsweise der Integration weniger Entwicklungsabteilungen, die am gleichen System unter verschiedenen Aspekten arbeiten. Die Komplexität lose gekoppelter Metamodelle ist bedeutend grö-

ßer und bedarf der Gestaltung von Konsistenzbedingungen. Dieser Ansatz sollte gewählt werden, wenn beispielsweise ein gesamtes Unternehmen den Austausch von Systemdaten über Bereichsgrenzen austauschen möchte.

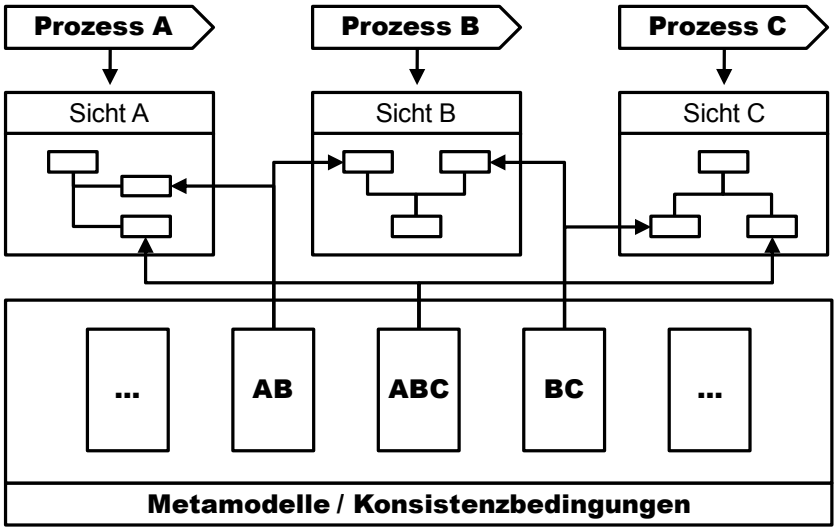


Abbildung 4-4 Ansatz der losen Kopplung
Quelle: GÜNZLER (2005, S.76), leicht modifiziert

Die Diskussion über den Segmentierungsansatz lässt die Definition der ersten Elemente eines Metamodells für Flugzeugsysteme zu. Abbildung 4-5 zeigt links die Grundelemente eines eng gekoppelten Metamodells und rechts die Elemente eines lose gekoppelten Metamodells in UML-Notation. Die schwarze Raute des Verbinders kennzeichnet eine *Komposition*, welche die Beziehung zwischen einem Ganzen und seinen Teilen beschreibt. Im konkreten Fall bedeutet die Komposition, dass ein System im Rahmen der engen Kopplung aus mindestens einer Sicht besteht (Multiplizität: 1..*) bzw. innerhalb der losen Kopplung aus mindestens einer Sicht, die über Relationen mit anderen Sichten verknüpft sein kann (Multiplizität: 0..*).

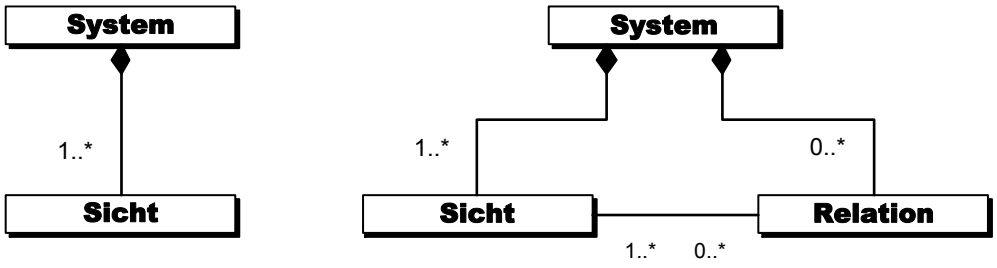


Abbildung 4-5 Eng und lose gekoppeltes Metamodell
(eigene Darstellung in Anlehnung an GÜNZLER, 2005; UML, 2011)

Für die Demonstration der Ziele dieser Arbeit wird aufgrund des begrenzten Datenaustauschvolumens zwischen einer geringen Zahl von Entwicklungsdomänen der Ansatz der engen Kopplung weiterverfolgt und demnach lediglich ein Metamodell zur Repräsentation des Systems innerhalb des Anwendungsbeispiels (Kapitel 6) erstellt, in dem

keine Konflikte zwischen den Partialmodellen auftreten. Für umfangreichere Problemstellungen sollte das Prinzip der losen Kopplung für eine potenzielle Anwendung diskutiert werden bzw. bestehende Modelle auf Basis der engen Kopplung durch das Hinzufügen von Konsistenzbedingungen erweitert werden.

4.4.2 Strukturierung

Ein technisches System besteht aus Elementen, die miteinander in Beziehung stehen und in ihrer Gesamtheit die Systemstruktur bilden. Eine Sicht kann dabei die gesamte Struktur betrachten oder einen Ausschnitt, der in einem bestimmten Kontext relevant ist. Dementsprechend ist innerhalb der Struktur des Metamodells jede Sicht um Elemente zu erweitern und die Möglichkeit zu schaffen, die Elemente über verschiedenartige Beziehungen zu verknüpfen. Abbildung 4-6 zeigt die erweiterte Darstellung des Metamodells wobei unter Berücksichtigung der Forderung nach Ausdruckskraft explizit die Modellierung einer beliebigen Anzahl von Element- und Beziehungstypen geschaffen werden muss. Jede Sicht besteht nach der UML-Notation aus mindestens einem Element (Multiplizität: 1..*), zwei Elemente können durch eine oder mehrere unterschiedliche Relationen verknüpft sein (Multiplizität: 0..*).

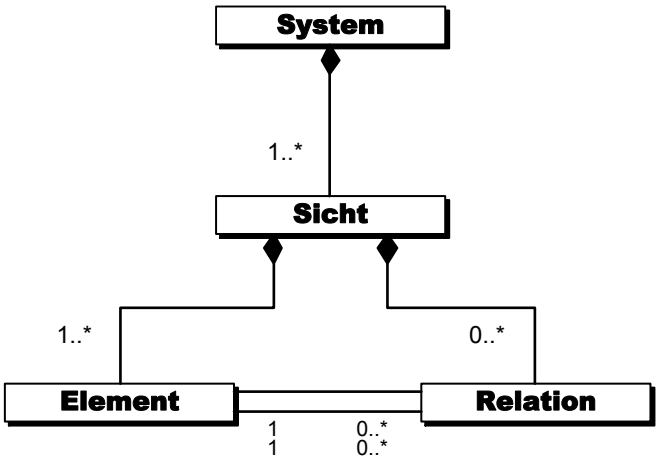


Abbildung 4-6 Einführung von Strukturaspekten in das Metamodell

4.4.3 Typisierung

Begriffliche Abgrenzung: In objektorientierten Sprachen wie der UML wird zwischen *Typen* und *Instanzen* unterschieden. Dabei sind Instanzen Modellierungselemente, die unterschiedliche Zustände einnehmen können und Typen Modellierungselemente, die einen gemeinsamen Zustandsraum von Instanzen definieren. Der Wertebereich von Instanzen eines Typs ist demnach identisch und es existiert zwischen Typen und Instanzen eine 1:0..* Relation (KATZKE, 2008). Weiterhin ist eine *Klasse* innerhalb der UML ein Typ, welcher Merkmale besitzt, die durch *Attribute* und *Operationen* be-

schrieben werden. Ein *Objekt* ist die Instanz einer Klasse und demnach wird die Erstellung eines Objekts *Instanziierung* genannt (UML, 2011).

Wie in nahezu allen Bereichen der rechnergestützten Entwicklung sollte die Erstellung von Systemmodellen durch die Definition von wiederverwendbaren Klassenelementen vereinfacht werden. Innerhalb von CAD-Systemen spricht man von feature-basierter Geometrieerstellung. Dabei werden dem Anwender häufig wiederkehrende Konstruktionselemente wie beispielsweise Quader oder Zylinder als parametrisierte Bausteine zur Verfügung gestellt, die durch Zuweisung von konkreten Werten als Modellinstanz erzeugt werden. Das Prinzip der Klassifizierung und Instanziierung über Datentypen ist beispielhaft in Abbildung 4-7 dargestellt.

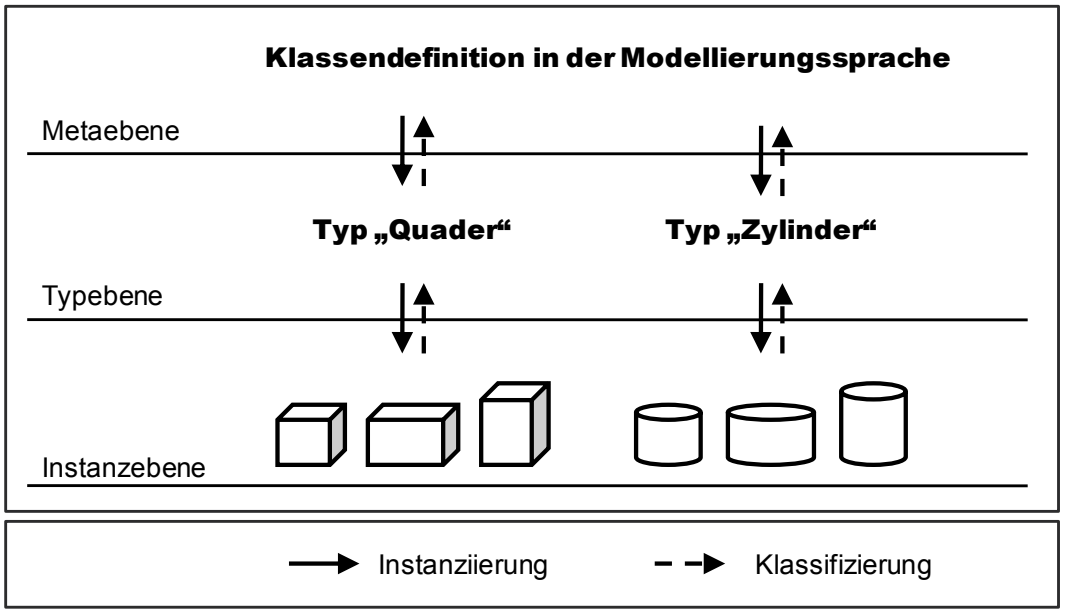


Abbildung 4-7 Prinzip der Instanziierung und Klassifizierung
Quelle: SCHICHTEL (2002), GÜNZLER (2005), ARNOLD ET AL. (2011)

Entsprechend des Beispiels aus dem Bereich der CAD-Modellierung ist das angestrebte Metamodell um Typen zu erweitern, welche die Wiederverwendung von einmal definierten, gemeinsamen Eigenschaften für die Erstellung ähnlicher Komponenten erlaubt. Abbildung 4-8 zeigt das um den Aspekt der Typisierung erweiterte Metamodell zur Beschreibung von Flugzeugsystemen. Eine Sicht beinhaltet also Systemelemente und deren Beziehungen, wobei jedes Element einem Typ zugewiesen wird. Die UML-Notation für diese Zuweisung wird durch einen Pfeil mit weißer Spitze gekennzeichnet und wird mit *Generalisierung* (auch: *Vererbung*, *Spezialisierung*, *Verfeinerung*) bezeichnet (KATZKE, 2008; UML, 2011). Jedem Typ können dabei beliebig viele Elemente zugewiesen werden, welche als Instanzen die konkrete Ausprägung dieses Typs darstellen.

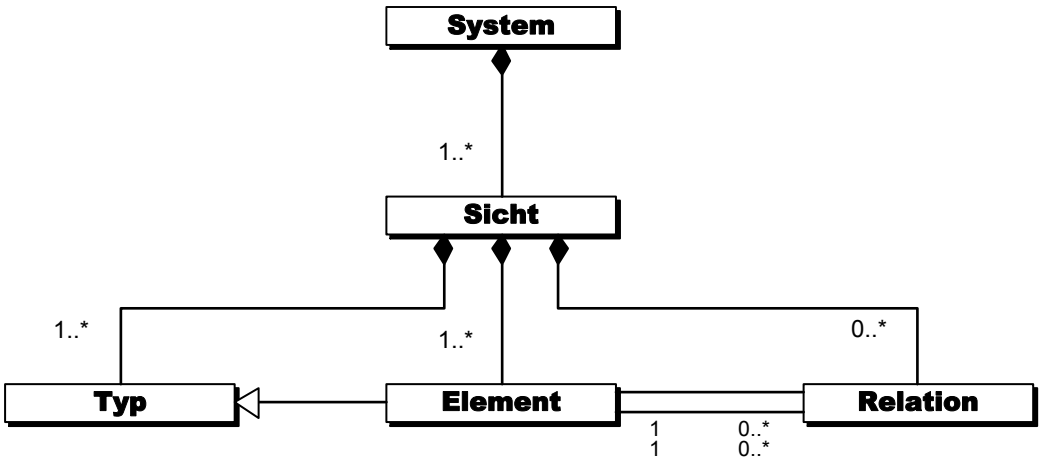


Abbildung 4-8 Einführung von Typaspekten in das Metamodell

4.4.4 Attributierung

Die Grundlage für die Klassifizierung von Systemen stellt die Verwendung von abstrakten Typen dar. Um die gleichartigen Eigenschaften einer Klasse von Systemen zu beschreiben, muss das Metamodell um diese Modellierungskonstrukte erweitert werden. Wie in Kapitel 4.4.3 bereits erwähnt, werden Klassen innerhalb der UML durch Attribute beschrieben und dieser Ansatz soll für das Metamodell dieser Arbeit übernommen werden.

Ziel der Attributierung ist es, alle Eigenschaften des betrachteten Systems über seinen Lebenszyklus hinweg beschreiben zu können. Für die Beschreibung von Flugzeugsystemen bedeutet dies beispielweise die Abbildung geometrischer Parameter von Kompressoren oder öffnungsabhängiger Durchflussraten von Ventilen. Abbildung 4-9 zeigt das um Attribute erweiterte Modell für Flugzeugsysteme, wobei die Eigenschaften eines jeden Typen durch eine beliebige Anzahl an Attributen exakt modelliert werden können. Natürlich besteht eine Vielzahl von verschiedenen Attributen und dementsprechend verschiedene Datentypen zum Zwecke der Beschreibung. Beispielsweise kann der Materialtyp eines Elements durch den Datentyp *string* beschrieben werden, wohingegen das drehzahlabhängige Kompressionsverhalten eines Verdichters durch einen Vektor vom Datentyp *real* beschrieben wird. Es wird im Folgenden davon ausgegangen, dass handelsübliche Werkzeuge zur Modellierung auf Basis von UML die Möglichkeiten zur Verwendung verschiedener Datentypen bereithalten. Deshalb wird der Aspekt der Modellierung von Datentypen als Generalisierung von Attributen nicht explizit in das Metamodell übernommen.

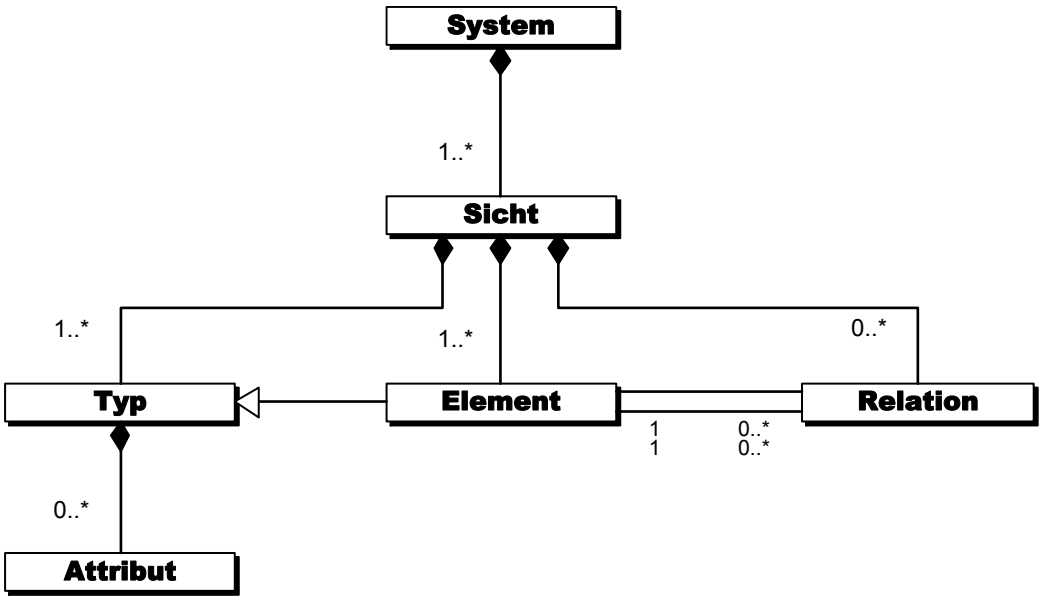


Abbildung 4-9 Einführung von Attributen in das Metamodell

4.5 Potenzielle Erweiterungen

Die vorangegangenen Kapitel 4.4.1 bis 4.4.4 beschreiben auf Basis logischer Anforderungen sowie Prinzipien der objektorientierten Programmierung ein Basiskonzept für die integrierte Modellierung von Flugzeugsystemen. Das in Abbildung 4-9 dargestellte Metamodell ermöglicht die Beschreibung von Systemen durch die Möglichkeit zur Erstellung von Systemsichten, Systemelementen und –strukturen sowie Elementtypen und deren Eigenschaften in Form von Attributen.

Das entworfene Metamodell entspricht damit den gestellten Anforderungen (vgl. Kapitel 4.3) und stellt die Basis für die Modellierung von Systemen dar. Aufgrund der Berücksichtigung objektorientierter Prinzipien ist das entworfene Metamodell trotz seiner geringen Zahl von Konstruktionselementen zur Beschreibung von komplexen Strukturen geeignet. Verschiedene Aspekte zur Erweiterung können jedoch für eine industrielle Anwendung oder spezielle Anwendungsfälle von Relevanz sein und werden hier ohne Anspruch auf Vollständigkeit erwähnt. Diese Erweiterungen sind jedoch für die Demonstration des modellbasierten Vorgehens im Rahmen dieser Arbeit nicht erforderlich.

Relationserweiterung: Relationen sind ein Konstrukt des Metamodells zur Beschreibung von Beziehungen zwischen Elementen. Bei einer vorhandenen Vielzahl verschiedener Relationen oder dem Wunsch nach Beschränkung der Anzahl von auswählbaren Relationen kann analog dem vorgestellten Prinzip der Typisierung das Metamodell um Relationstypen erweitert werden. Die Einführung von Relationstypen

ermöglicht auch die Wiederverwendbarkeit von einmal definierten Relationen und kann somit auch die Modellerstellung beschleunigen.

Relationsattribute: Im Rahmen der Erweiterung des Metamodells um Relationstypen würde die parallele Einführung von Relationsattributen vorteilhaft sein. So könnten Relationstypen in ganzheitlicher Weise mit Attributen versehen werden und somit ebenfalls die Modellerstellung beschleunigen.

Typenerweiterung: Analog der Relationserweiterung kann bei komplexen Modellen, welche aus einer Vielzahl unterschiedlicher Elementtypen bestehen, die Erweiterung des Metamodells gemäß dem Prinzip der Typisierung im Bereich der Typen sinnvoll sein. Die Einführung verschiedener Typenklassen würde eine bessere Strukturierung erlauben und damit die Übersichtlichkeit verbessern.

4.6 Zusammenfassung

Die Modellbildung ist ein unerlässliches Hilfsmittel innerhalb der frühen Phasen von Entwicklungszyklen geworden und basiert auf dem Prinzip der Abstraktion. Abstrahierte Systeme werden unter einem gewissen Blickwinkel betrachtet und weisen dementsprechend eine geringere Zahl an relevanten Parametern auf, als dies bei einer ganzheitlichen Betrachtung der Fall wäre. Dies bedeutet jedoch für alle beteiligten Einzeltätigkeiten eine unterschiedliche Sicht auf das gleiche System. In der Folge wird in den Entwicklungsdomänen eine Vielzahl von Partialmodellen mit einer ebenso großen Bandbreite an Werkzeugen erstellt und bearbeitet. Die Verwendung einer großen Bandbreite an Werkzeugen wurde jedoch nicht als Hindernis identifiziert sondern stellt im Gegenteil eine Prämisse für den erfolgreichen Systementwurf dar (vgl. Kapitel 2.4.3).

Aufgrund der geänderten Bedingungen in denen der Entwurf von technischen Systemen abläuft (vgl. Kapitel 2.1), müssen Entwicklungsvorgänge stärker parallelisiert und die Koordination zwischen Entwicklungsdomänen verbessert und automatisiert werden. Eine Möglichkeit ist der integrierte Systementwurf auf Basis von Metamodellen, die bereichsübergreifend die Entwicklung von Partialmodellen steuern und somit die Erfolgsfaktoren zukünftiger Systementwicklungsmethoden (vgl. Kapitel 3.5) berücksichtigt. Dabei erfasst ein Metamodell alle Eigenschaften der verwendeten Partialmodelle, die für die Modellerstellung oder –evaluation innerhalb anderer Domänen benötigt werden. Das Metamodell dient weiterhin zur Koordination der Partialmodelle über die Phasen eines Lebenszyklus hinweg und bildet damit die Basis für die Steuerung von Entwicklungstätigkeiten.

Die Erstellung eines allgemeingültigen Modells für die Beschreibung von Flugzeugsystemen setzt grundlegende Überlegungen über die Inhalte und Funktionen des Systemmodells voraus. Deshalb wurde das Metamodell basierend auf den vier Prinzipien Segmentierung, Strukturierung, Typisierung und Attributierung aufgebaut, um die gewünschte Ausdruckskraft bei gleichzeitiger Berücksichtigung der Umsetzbarkeit zu erreichen. Aufgrund der im Entwurf technischer Systeme vorhandenen Sichten, muss ein integriertes Metamodell segmentiert werden können, um diesen Anspruch abbilden zu können. Die Basis des Metamodells stellt also die Definition aller relevanten Sichten auf ein System dar. Für diese Darstellung sind dem Anwender entsprechende Modellierungskonstrukte (Elemente und Relationen) zur Verfügung zu stellen, um jedwede Systemstruktur abbilden zu können. Unter dem Aspekt der Typisierung von Modellen und der damit verbundenen Wiederverwendbarkeit werden Elementtypen eingeführt, die eine Einteilung in verschiedene, oft verwendete Elementklassen erlauben. Um schließlich die Systemeigenschaften ausdrucksstark abbilden zu können, werden Attribute zur parametrisierten Beschreibung von Elementen eingeführt.

Das in Abbildung 4-9 dargestellte Metamodell für Flugzeugsysteme ist das Ergebnis einer luftfahrtspezifischen Anforderungsanalyse für die integrierte Systementwicklung und baut auf Prinzipien auf, die aus der Informatik bekannt sind. Das hier vorgestellte Metamodell kann als Basiskonzept für Studien der integrierten Systementwicklung angesehen werden und mit den Erweiterungsvorschlägen (vgl. Kapitel 4.5) auf komplexere Fragestellungen erweitert werden. Für die vorliegende Arbeit und die Demonstration der Anwendbarkeit ist der hier vorgestellte Umfang des Metamodells jedoch hinreichend ausdrucksstark.

5 IMPLEMENTIERUNG DES METAMODELLS

Im letzten Kapitel wurden die Grundlagen für eine abstrakte und rein formale Strukturierung von Systemen diskutiert und daraus ein Metamodell zur Systembeschreibung abgeleitet. Ziel der Implementierung ist es, eine durchgängige Prozess- und Werkzeugkette von der konzeptionellen Beschreibung bis hin zur logischen und physischen Modellierung von Systemen aufzubauen. Die resultierende Infrastruktur aus Software- und Hardwareelementen stellt die Basis für die modellbasierte Entwicklung von Luftfahrtsystemen dar, was an einem Beispiel im Folgekapitel demonstriert wird.

5.1 Beschreibung der Infrastruktur

Das Metamodell aus Kapitel 4 stellt die theoretische Basis für die Implementierung eines Modells für Flugzeugsysteme dar. Das Metamodell beschreibt auf Basis der enthaltenen Modellierungskonstrukte Informationen verschiedener, domänenspezifischer Modelle, die zu diesem Zweck nach dem Black Box Prinzip (WALTER, 2010) abstrahiert werden müssen. Für jede Entwicklungsdomäne ist ein Metamodell zu definieren, welches die domänenübergreifenden Informationen abbildet. Durch die Definition der Struktur des Metamodells kann eine Transformationsdefinition festgelegt werden, um domänenübergreifende Informationen eines domänenspezifischen Partialmodells mit Hilfe eines Transformationsskripts in das Metamodell zu schreiben. Dieses Vorgehen (Abbildung 5-1) entspricht der Transformation von Modellen nach CZARNECKI UND HELSEN (2006).

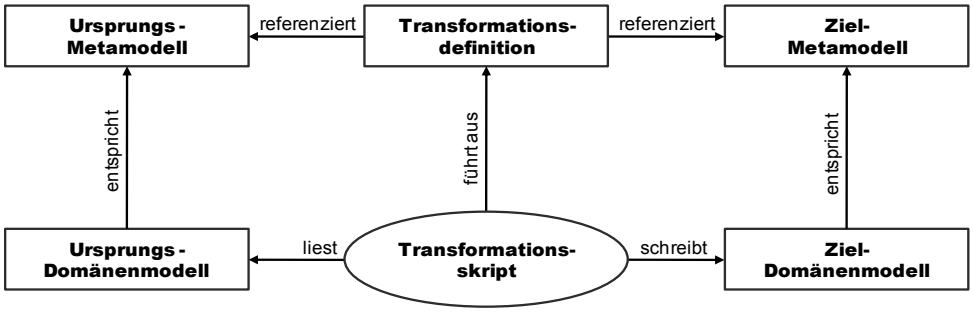


Abbildung 5-1 Konzept der Modelltransformation
Quelle: CZARNECKI UND HELSEN (2006), leicht modifiziert

Es gilt nun eine Infrastruktur zu definieren, die eine Abbildung des Metamodells innerhalb des Rahmens der aktuellen Entwicklung von Flugzeugsystemen (vgl. Kapitel 2) ermöglicht und dabei die Erfolgsfaktoren zukünftiger Systementwicklung berücksichtigt (vgl. Kapitel 3). Weiterhin ist den Anforderungen der Metamodellierung (vgl. Kapitel 4.3) zu entsprechen, um die Akzeptanz der modellbasierten Systementwicklung nicht zu behindern und das Potenzial des Vorgehens ausschöpfen zu können. Folgende Prämissen sollen deshalb nachfolgend gelten:

- Explizit zu verknüpfen und im Metamodell darzustellen sind:
  - Anforderungen und Funktionen
  - Funktionen und Systemelemente
  - Produkt und Organisation (Prozesse, Menschen)
- Fähigkeit zur Integration domänenspezifischer Werkzeuge
- Integrationsmöglichkeit in bestehende PDM Systeme
- Vermeidung von redundanten Daten
- Vermeidung der Einführung mehrerer Softwarewerkzeuge zur Lösung eines Problems

5.2 Metamodellierung

Für die Modellierung von technischen Produkten ist vor allem ein Ansatz bekannt, der unter dem Akronym STEP (Standard for the Exchange of Product Data) auf Basis der Normenreihe ISO 10303 (*Industrial Automation Systems and Integration – Product Data Representation and Exchange*) entstanden ist. Das anfänglich auf CAD-Daten spezialisierte Format wird kontinuierlich weiter ausgebaut mit dem Ziel, alle Produktdaten eines Produktlebenszyklus austauschen zu können (PRATT, 2001). Das STEP-Format baut auf mehreren Bausteinen auf, die es dem Anwender erlauben, anwendungsspezifische Produktmodelle zu erstellen. Um die umfangreichen Bausteine von STEP zu strukturieren, bestehen verschiedene Serien, die zum einen den Aufbau des Datenformats und Programmierschnittstellen beschreiben, zum anderen generelle Produktinformationen (*Integrated Resources*) oder spezifische Produktinformationen (*Application Protocols*) abbilden. Die *Integrated Resources* stellen den Kern der STEP-Produktmodelle dar und beschreiben die Geometrie und generelle Produktinformationen. Darauf aufbauend ergänzen die *Application Protocols* die Produktinformationen um anwendungs- oder branchenspezifische Daten. In Tabelle 5-1 sind einige der STEP-Bausteine sowie zugehörige Klassen aufgelistet.

Tabelle 5-1 Aufbau des STEP Datenformats; Quelle: ISO (1994)

Klasse	Serie	Beschreibung
Umfang und Architektur	1	Überblick und fundamentale Prinzipien
Beschreibungsmethoden	11	Definition der Sprache EXPRESS
Implementierungsmethoden	21	Format von Austauschdateien (STEP Physical File)
	23	C++ Language Binding
	25	EXPRESS to XMI binding
	27	Java Language Binding
	28	XML representations of EXPRESS schema and data
Testmethodik (31-40)	31	Generelle Konzepte zur Konformitätsprüfung
Integrated Resources (41-50)	41	Generelle Information zur Produktbeschreibung
	42	Darstellung der Geometrie
Application Protocols (201-299)	203	Darstellung konfigurierbarer Geometrie
	212	Elektrotechnische Informationen
	214	Produktlebenszyklus in der Automobilindustrie
	242	Managed Model Based 3D Engineering (in der Entwicklung)

STEP bietet also grundsätzlich die Möglichkeit, anwendungsspezifische Daten über ein Anwendungsprotokoll auszutauschen, setzt allerdings die allgemeine Akzeptanz aller Prozessbeteiligten und die Unterstützung der eingesetzten Rechenwerkzeuge voraus. Die ISO 10303 stellt für spezielle Einsatzzwecke – wie der Definition eines luftfahrtspezifischen Anwendungsprotokolls – ein Vorgehen zur Erstellung eines Anwendungsprotokolls zur Verfügung. Das Vorgehen basiert auf der Definition aller Prozesse des Anwendungsbereiches und deren Unterteilung in diskrete Aktivitäten geeigneter Größe. Danach werden alle Ein- und Ausgangsgrößen der Aktivitäten sowie benötigte Hilfsmittel bestimmt und die Aktivitäten untereinander durch Informations- und Materialflüsse verbunden. Über mehrere Zwischenmodelle werden in der Sprache EXPRESS-G schließlich die identifizierten Datenklassen und ihre Beziehungen modelliert und als Metamodell in Form eines Anwendungsprotokolls bereitgestellt.

STEP-Datensätze werden auf Basis eines ASCII-basierten Syntax ausgetauscht, der in der Serie 21 beschrieben wird. Dieser Ansatz erschwert die Interpretation der Modelle durch den Menschen und erfordert Software, die den Ansatz unterstützt. In der Regel erfolgt diese Unterstützung ausschließlich innerhalb von geometriebasierten CAD-Systemen und nicht in Berechnungs- und Simulationswerkzeugen bzw. integrierenden Systems Engineering Plattformen. Um die unternehmensweite Anwendbarkeit von STEP zu erreichen, wurden Transformationsschnittstellen zu XML (ISO (2007)) und UML (ISO (2005)) geschaffen, wobei beide noch nicht den Status eines internationalen Standards besitzen. So beschreiben LERCHER (2008) und SHEA (2011) – unter Erwähnung einiger akademischer Anwendungsbeispiele zur unternehmensweiten Datenintegration – die zögerliche Anwendung von STEP in der Industrie. Für diese Arbeit wurde deshalb die Betrachtung des STEP-Ansatzes für die generische Modellierung von Flugzeugsystemen nicht weiterverfolgt.

Für die theoretische Modellbildung von Systemen existiert weiterhin eine Vielzahl von Modellierungsansätzen, die ihren Ursprung zumeist in der Informatik haben. Ein allgemein einsetzbarer Ansatz ist das *Entity-Relationship-Model (ERM)*, welches auf den Entitäten (engl.: entity) und deren Beziehungen (engl.: relationship) beruht. Dabei stellen Entitäten voneinander abgrenzbare Gegenstände dar, die sich durch charakteristische Attribute beschreiben lassen (CHEN, 1976). Seit mehreren Jahren wird in der praktischen Anwendung neben dem Entity-Relationship-Modell auf objektorientierte Modellierungstechniken wie die *Unified Modeling Language* (UML) gesetzt (OESTERREICH, 2001; OMG, 2011). Die grafische Beschreibung von Objekten innerhalb verschiedener Diagrammtypen bietet einen weitaus größeren Funktionsumfang und hat sich über viele Anwendungsbereiche hinweg etabliert. Die *Systems Modeling Language* (SysML) stellt als Weiterentwicklung von UML Modellierungskonstrukte zur Verfügung, welche speziell auf die Systementwicklung zugeschnitten ist. Durch entsprechende Semantik sowie vordefinierte Diagrammtypen und Modellbibliotheken wird der Funktionsumfang der UML an die Bedürfnisse des Systementwicklers angepasst und somit die Anwendung erleichtert. GÜNZLER (2005) weist weiterhin auf Semantische Netze und Ontologien als allgemeine Möglichkeiten der Modellbildung auf Metaebene hin.

Für die Modellierung technischer Systeme scheint der auf UML basierende Standard *Meta Object Facility* (MOF) (MOF, 2011) geeignet, der durch die *Object Management Group* (OMG) beschrieben wird (OMG, 2011) und das Problem des Informationsaustausches zwischen verschiedenen Domänen explizit behandelt. Die Modellierungstechnik der OMG eignet sich für die Beschreibung von Modellen und ihren Instanzen in beliebigen Anwendungsbereichen. Mit Blick auf das Thema Datenintegration ist es notwendig, Modellinstanzen verschiedener Modellierungssprachen ineinander zu überführen oder zumindest relevante Ausschnitte in die jeweils andere Sprache zu überführen. Um dies zu gewährleisten, müssen die Metamodelle der unterschiedlichen Sprachen untereinander kompatibel sein und aufeinander aufbauen (vgl. Abbildung 5-2).

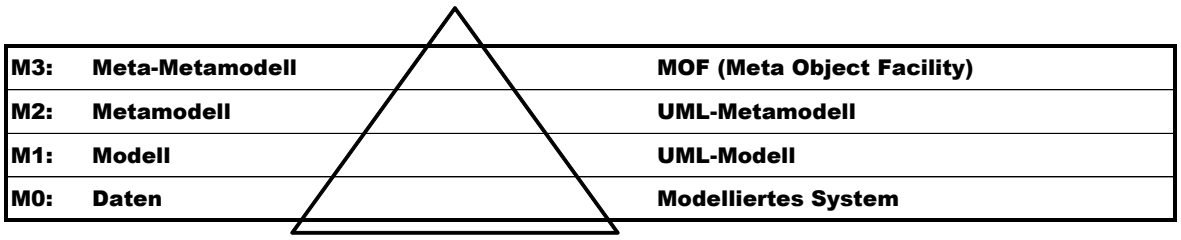


Abbildung 5-2 Modellierungsebenen der OMG Metamodellarchitektur
Quelle: eigene Darstellung nach OMG (2006)

In Analogie dazu zeigt Abbildung 5-3 die Modellierungsebenen, die für das Metamodell von Flugzeugsystemen notwendig sind. Basis sind die verschiedenen Partialmodelle, die in den beteiligten Domänen entworfen werden (z.B.: CAD-Modell). Diese müssen in ein Metamodell überführt werden, was die domänenübergreifenden Informationen abbildet. Aufgrund der Vielzahl der im Entwurf komplexer Systeme beteiligten Werkzeuge bietet sich die Erstellung von mehreren Metamodellen an, die je ein Partialmodell abbilden. Schließlich werden alle Metamodelle in ein integrierendes Metamodell vereint, das alle domänenübergreifenden Informationen eines Systems beinhaltet. Dieses Metamodell wird wiederum durch das Meta-Metamodell beschrieben, welches eine formale Beschreibung der Systemstruktur darstellt.

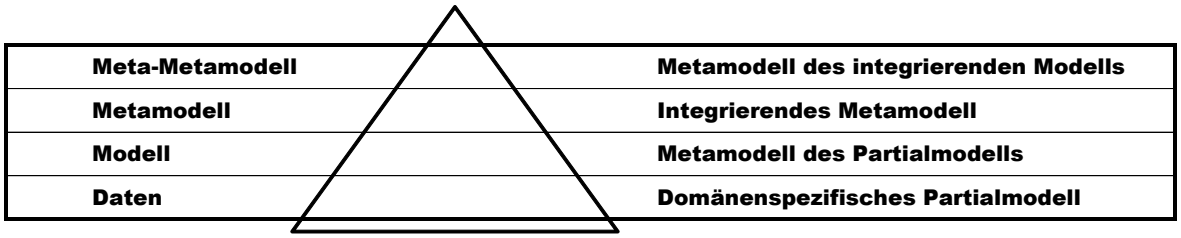


Abbildung 5-3 Modellierungsebenen Metamodell Flugzeugsysteme

Das Prinzip verschiedener Modellierungsebenen basiert auf dem Prinzip der Abstraktion, wobei man die verschiedenen Ebenen als schrittweise Verfeinerung vom Abstrakten zum Konkreten betrachten kann. LERCHER (2008) beschreibt in diesem Zusammenhang den Einsatz von Abstraktionsebenen als Trennung zwischen abstrakten *Typen* und konkreten *Instanzen*, die sich in objektorientierten Programmiersprachen findet. Ein einmal definiertes Metamodell kann deshalb immer wieder für die Beschreibung eines Systems verwendet werden. Durch die Belegung der vorgesehenen Freiheitsgrade (*Instanziierung*) kann der Entwickler die eigentlichen Modelle des Systems als eine konkrete Version erzeugen.

Um der Anforderung der Handhabbarkeit (vgl. Kapitel 4.3) zu entsprechen, sollte auf bereits etablierten Metamodellen aufgesetzt werden. Aufgrund der weiten Verbreitung der UML (vgl. Kapitel 3.3.2) in Forschung und Industrie, bietet sich die Verwendung des UML-Metamodells an, das eine Ausprägung des MOF darstellt und sich zur Erstellung von Modellen und Metamodellen jeglicher Art eignet.

Auch die Implementierung des Metamodells auf Basis von SysML bietet sich an, da der Funktionsumfang der UML speziell auf die Bedürfnisse von Entwicklern (technischer) Systeme zugeschnitten ist und die Anwendung durch die graphische Orientierung deutlich erleichtert wird. So schlagen SJÖSTEDT (2009) und QAMAR (2011) die Verwendung eines SysML-Modells vor, das über das Eclipse Modeling Frame-

work (EMF) mit den domänenspezifischen Werkzeugen kommuniziert. Der offensichtliche Nachteil der Verwendung von EMF als Metamodellierungsplattform ist allerdings die Komplexität der Umsetzung und die Abhängigkeit von lizenzpflichtigen Programmen.

Auf Basis der genannten Alternativen (ERM, UML, SysML), wird die Infrastruktur des Metamodells mit SysML modelliert. Zum einen ist der Funktionsumfang der UML auf die Bedürfnisse der Systementwickler bereits zugeschnitten und die Anwendung wird durch die graphische Orientierung und ein großes Maß an Standardisierung der Sprachelemente deutlich erleichtert. Zum anderen werden bereits diskutierte Ansätze des Systementwurfs (z.B.: DoDAF, MODAF) durch spezielle Bibliotheken unterstützt (vgl. Kapitel 3). Weiterhin erfüllt die System Modeling Language die Anforderung nach durchgängiger Abbildung von Anforderungen, Funktionen und Endprodukt und ist ohne die Einführung einer Vielzahl von Werkzeugen umsetzbar.

Die bereits erwähnten, vordefinierten Diagrammtypen schränken den Modellierer in seiner Flexibilität nicht ein, sondern geben lediglich den Rahmen der Modellierungstätigkeiten vor. Der Systementwickler legt die Anforderungen an ein System durch *Requirement Diagram* (*req*) und *Use Case Diagram* (*uc*) fest. Dabei werden explizit die Zielvorgaben der Funktionalen Produktentwicklung (Kapitel 3.1.2) übernommen. Die Definition der physischen Eigenschaften von Komponenten erfolgt in *Block Definition Diagram* (*bdd*), *Internal Block Diagram* (*ibd*) und *Parametric Diagram* (*par*). Die Möglichkeit zur Verknüpfung von Anforderungen mit Komponenten erhöht die Transparenz in der Synthesephase der Entwicklung. Beispielsweise kann eine physische Komponente mit einer Anforderung durch die Beziehung *satisfy* verknüpft werden. Verschiedene graphische und tabellarische Auswertungsmöglichkeiten wie die Erstellung von Adjazenzmatrizen lassen weitere Analysen wie die Clusteranalyse zur Erkennung modularer Produktstrukturen zu. Die Funktionen eines Systems werden durch *Activity Diagram* (*act*), *State Machine* (*stm*) und *Sequence Diagram* (*sd*) beschrieben. Aktivitätsdiagramme beschreiben den Ablauf von Daten- und Stoffflüssen innerhalb eines Systems und erlauben eine weitere Detaillierung der Systemdefinition. Weiterhin stellen *act*-Diagramme oft die Basis für *uc*-Diagramme dar, welche verschiedene Anwendungsfälle eines Systems definieren. Im Allgemeinen werden *Use Cases* für die Absicherung verschiedener Anforderungen definiert und verwendet.

Für das hier vorgestellte Vorgehen wird nur ein Teil der SysML-Funktionalität verwendet. Im Speziellen sind dies *req*-, *uc*-, *bdd*,- *ibd*- und *ac*-Diagramme. Diese Diagrammtypen unterstützen die Synthesephase der Produktentwicklung und damit die Erstellung von konzeptionellen Systemstrukturen. Durch die Zuordnung von Anforde-

rungen und Komponenten (engl.: mapping) sowie der Definition von *uc*-Diagrammen werden bereits anforderungsgerechte Systemstrukturen erzeugt, die in nachfolgenden Phasen analysiert werden sollen.

Durch die Erstellung der beschriebenen SysML-Diagramme entsteht eine vollständige, interaktive und graphische Systembeschreibung. Weiterhin bietet SysML die Möglichkeit, eine anwendungsspezifische Modellierung vorzunehmen ohne auf Datenformate mit fest definierten Datenstrukturen zurückgreifen zu müssen. Die Modellierung von Systemen und den zugehörigen Prozessen mit SysML erlaubt somit eine durchgängige, evolutionäre Entwicklung von Systemen unter Berücksichtigung der aktuellen Methoden der Systementwicklung (vgl. Kapitel 3). Die Verwendung von SysML und entsprechender Diagrammtypen innerhalb des Vorgehensmodells nach VDI 2206 ist in Abbildung 5-4 dargestellt.

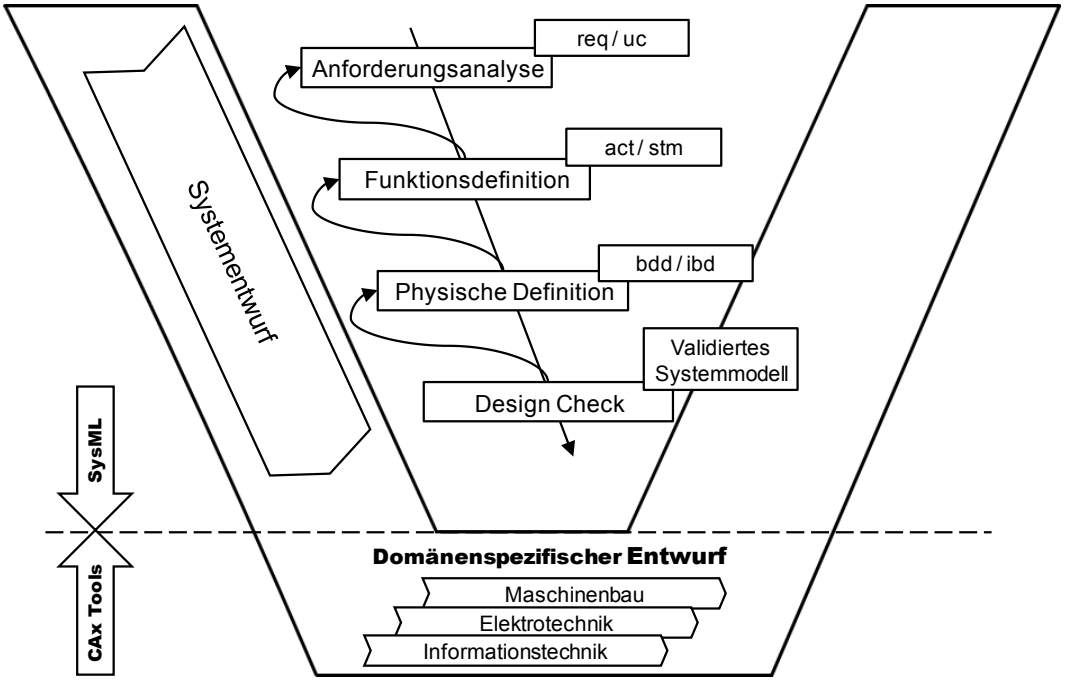


Abbildung 5-4 SysML-Diagramme im Systementwurf
Quelle: in Anlehnung an VDI (2004), KOSSIAKOFF ET AL. (2011)

5.3 Datenbankstruktur

Eine direkte Verknüpfung zwischen SysML-Modell und weiteren Werkzeugen ist, wie in Kapitel 5.2 diskutiert, nur über ein Metaebene möglich. Diese Metaebene kann, wie bereits angedeutet, mittels des EMF umgesetzt werden, indem Code zur automatisierten Übersetzung zwischen den Metamodellen geschaffen wird. Eine andere Möglichkeit bietet die dateibasierte Darstellung der Metaebene, wobei das XML-Format hierfür eine Möglichkeit darstellt. Zusätzlich zu dateibasierten Verfahren bieten sich Da-

tenbanken an, welche folgende Vorteile gegenüber Dateien bieten (SAAKE, HEUER UND SATTLER (2008, S.7 ff.)):

- Gleichzeitiger Zugriff von verschiedenen Benutzern (Mehrbenutzerfähigkeit)
- Konsistenzüberwachung der Datenbankinhalte
- Unterstützung von redundanzfreier / redundanzarmer Speicherung
- Zugriffskontrolle durch Benutzerrollen und Rechte

Ein weiterer Vorteil einer Datenbankstruktur ist die einfache Anbindung jeglicher Werkzeuge, die von Ihnen erzeugte Daten in nicht proprietären, unverschlüsselten Dateiformaten abspeichern können. Viele Werkzeuge bieten bereits implementierte Schnittstellenfunktionalität zu SQL-Datenbanken an; Werkzeuge, die textuelle Ausgaben jeder Art unterstützen, können über einfache Postprozessoren ebenso an Datenbanken angebunden werden.

Zur Speicherung von Daten in Datenbanken sind für den Anwendungsbereich dieser Arbeit mehrere Möglichkeiten denkbar. Das zentrale Ziel der Verwendung einer Datenbank soll die Entkoppelung von Daten und den auf sie zugreifenden Programmen sein (VOSSEN, 2008). Wie bereits in Kapitel 2.4.2 erwähnt, stehen heute vor allem Relationale Datenbanken (RDB), Objektorientierte Datenbanken (OODB), Objektrelationale Datenbanken (ORDB) und XML-Datenbanken zur Auswahl. Diese werden von kommerziellen Anbietern und Open Source Projekten angeboten und unterscheiden sich unter anderem durch die Funktionalität des Datenbank Management Systems (DBMS) sowie ihren Preis.

Relationale Datenbanken stellen heute einen de-facto-Standard für zahlreiche kommerzielle Anwendungen dar (VOSSEN, 2008). Daten werden in Tabellen (sog. Relationen) abgelegt, wobei die Spalten einer Tabelle die Attribute des Datensatzes darstellen. In den Zeilen (Tupel) werden die unabhängigen Datensätze (engl.: record) abgelegt. Größere Datensätze werden nach logischen Aspekten auf mehrere Tabellen aufgeteilt und über sogenannte Keys verknüpft (SHEA, 2011). Der Umfang der möglichen Datentypen ist allerdings auf einfache Typen beschränkt (*Integers, Real, Text, Binary, Temporal, Spatial, Set of Bits*) und man besitzt – im Vergleich zu höheren Programmiersprachen – meist keine Möglichkeit aus den vorhandenen Typen neue zu erzeugen oder neue Typen einzuführen (VOSSEN, 2008). RDB eignen sich demnach gut für die Speicherung und Verwaltung von großen Mengen an einfach strukturierten Daten.

Objektorientierte und objektrelationale Datenbanken orientieren sich an den wachsenden Bedürfnissen der Anwender. Die aus modernen Anwendungen auf die Datenbank abzubildenden Objekte besitzen gerade in technischen Anwendungen (CAx-Systeme)

eine komplexe Struktur. Diese lassen sich mit den „flachen“, insbesondere *record*-orientierten Datenbanken nur unzureichend beschreiben (VOSSEN, 2008). In OODB und ORDB wurden die von Programmiersprachen bekannte Objektorientierung mit modernen Datenbanktechniken vereint, um erweiterbare Datenbanken zu realisieren, Programmiersprachen und Datenbanksprachen zu integrieren und Datenverteilung in heterogenen Systemen zu realisieren (VOSSEN, 2008). OODB und ORDB sollten dort Einsatz finden, wo komplex strukturierte und heterogene Daten verwendet werden.

XML-Datenbanken (XMLDB) gehören den semistrukturierten Datenbanken an und unterscheiden sich von klassischen Datenbanktypen darin, dass sie kein fest vorgegebenes Schema benutzen. Dies äußert sich vor allem darin, dass die Menge der Attribute von Einheiten gleichen Typs variieren kann. Durch das Fehlen einer Struktur ist für das Auffinden eines Datensatzes die Selbstbeschreibung des Datensatzes notwendig. In XMLDB werden diese Daten explizit durch sogenannte *Tags* beschrieben (Bsp.: `<tag>data</tag>`). Die hierarchische Struktur von Daten wird durch die Verschachtelung von Tags erreicht. XML-Datenbanken werden beispielsweise in Datenintegrationsszenarien verwendet, in welchen Daten aus unterschiedlichen Quellen zu einem einheitlichen Bestand zusammengefasst werden sollen (VOSSEN, 2008).

Welcher Datenbanktyp für die Implementierung in einer Entwicklungsumgebung gewählt wird, hängt neben den oben diskutierten Kriterien von einer Vielzahl von Faktoren ab. Dazu gehören die Menge und Art der zu speichernden Daten, die Zugriffshäufigkeit oder die Zahl der Benutzer.

Tabelle 5-2 Vergleich von Datenbanktypen

	RDB	OODB	ORDB	XMLDB
Große Akzeptanz / weite Verbreitung	☐			
Open Source DBMS verfügbar	☐			
Standalone- Installation möglich	☐			
Schnittstellen für Datenaustausch vorhanden	☐			☐
Metamodell durch Datentypen repräsentiert	☐	☐	☐	☐
Anwendbarkeit Datenbanktyp	☐	☐	☐	☐
Implementierungsaufwand	☐			
Semantische Ausdrucksfähigkeit		☐	☐	☐
Abbildung komplexer Datenstrukturen		☐	☐	☐

Eine detaillierte Analyse würde den Rahmen dieser Arbeit sprengen und bleibt dem industriellen Anwender überlassen. Für das industrienahe Anwendungsbeispiel dieser Arbeit (vgl. Kapitel 0) wurde eine Relationale Datenbank (DBMS: MySQL & GUI: HeidiSQL) aufgrund der in Tabelle 5-2 gezeigten Kriterien ausgewählt.

Zunächst sind die weite Verbreitung des Datenbanktyps und offene Datenbankstandards für eine Verwendung ausschlaggebend. Weiterhin wurde eine relationale Datenbasis (RDB) gewählt, da diese mit Open-Source-Programmen in Desktopumgebungen zu implementieren ist und durch klar strukturierte Syntax eine schnelle Schnittstellengestaltung ermöglicht. Ein weiterer Aspekt von relationalen SQL-Datenbanken ist die potenzielle Verwendung der Datenbankeinträge innerhalb von Webseiten. Beispielsweise können die bereits erfassten und systematisierten Anforderungen an ein System im Rahmen einer Weboberfläche allen Beteiligten zur Verfügung gestellt werden und als zentrales Element des Anforderungsmanagements dienen.

Die weite Verbreitung des relationalen Datenbankmanagementsystems MySQL für Webanwendungen hat zur Anwendung von MySQL auch in technisch orientierten Branchen geführt. Dieser Umstand und die freie Verwendbarkeit haben zu einer Vielzahl von Schnittstellen zwischen Softwareanwendungen und relationalen Datenbanken geführt. Erwähnenswert ist die Java-Database-Connectivity (JDBC) Schnittstelle, welche u.a. in der Berechnungssoftware Matlab[7] und dem Modellierungswerkzeug MagicDraw[8] verwendet wird, um Datenbanken zu manipulieren. Unter Verwendung dieser Schnittstelle ist beispielsweise eine direkte Übersetzung der Struktur des SysML-Metamodells in die Struktur einer relationalen Datenbank möglich (MagicDraw). Die Eigenschaften der Systemkomponenten werden in die Datenbanksprache SQL übersetzt und über die JDBC-Schnittstelle mit der Datenbank kommuniziert (vgl. Abbildung 5-5). Auch die rekursive Änderung der Datenbankstruktur in SysML kann so abgebildet werden.

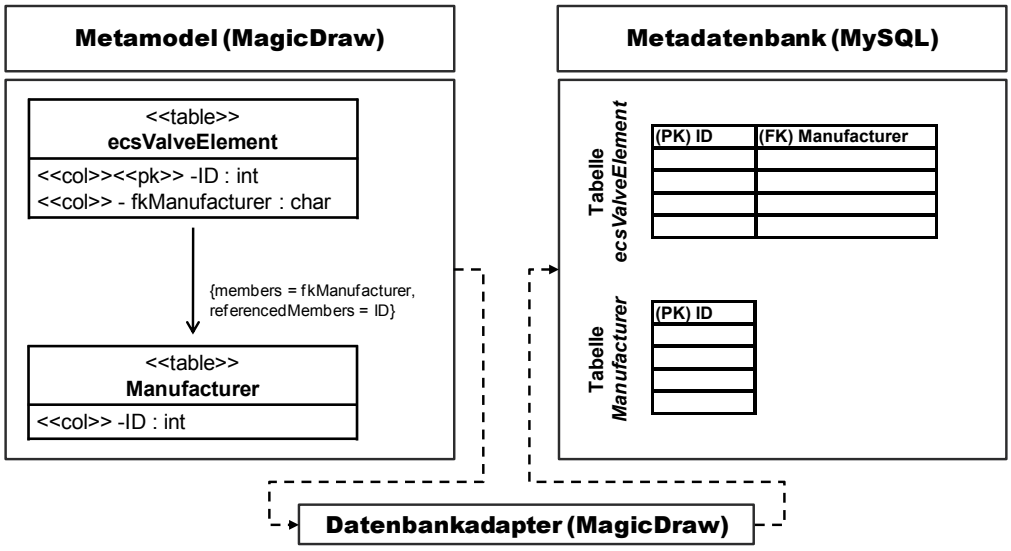


Abbildung 5-5 Datenbankmodellierung mit MagicDraw

[7] www.mathworks.de
[8] www.magicdraw.com

Die Repräsentationsmöglichkeit des Metamodells in der Datenbank muss ebenfalls gewährleistet sein. Abbildung 5-6 zeigt beispielhaft die Erzeugung einer Klasse innerhalb von SysML unter Berücksichtigung der Struktur des Meta-Metamodells (vgl. Kapitel 4). Weiterhin wird die Erzeugung eines Objektes (Instanziierung des Metamodells) gezeigt, das in der relationalen Datenbank abgelegt werden kann.

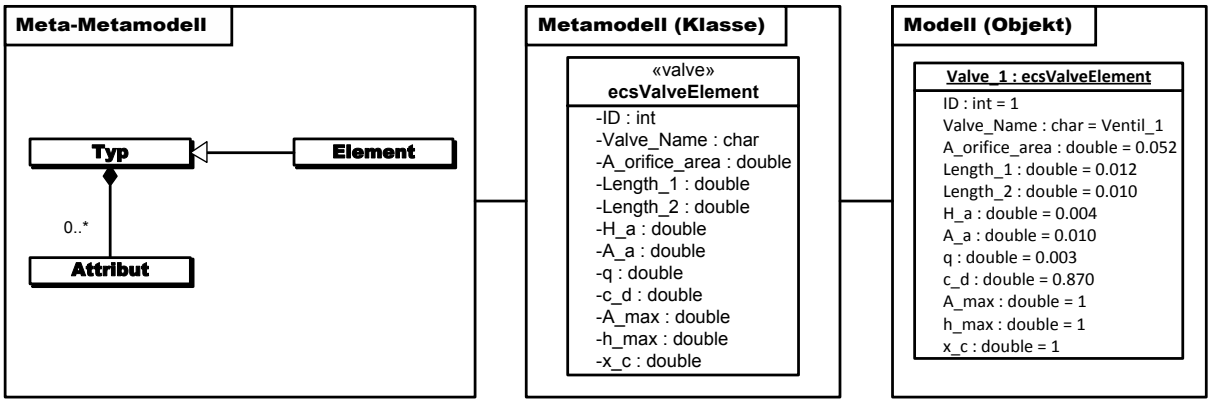


Abbildung 5-6 Instanziierung des Metamodells

Die semantische Ausdrucksfähigkeit sowie die Möglichkeit zur Abbildung komplexer Datenstrukturen sind ebenfalls Leistungsmerkmale von Datenbanken. Diesen Eigenschaften sind besonders in OODB, ORDB und XMLDB zu finden, resultieren jedoch in erhöhtem Implementierungs- und Anwendungsaufwand. Aus dem zu erwartenden Modellierungsumfang. Für den Einsatzzweck dieser Arbeit sind vielmehr eine Datenbanksprache mit adäquatem Umfang und etablierte Schnittstellen der Datenbank zu anderen Applikationen wichtig, was die Wahl einer relationalen Datenbank begründet.

Abbildung 5-7 zeigt den resultierenden Aufbau der gewählten Infrastruktur für die Etablierung eines Metamodells für Flugzeugsysteme, die in den folgenden Abschnitten genauer erläutert wird.

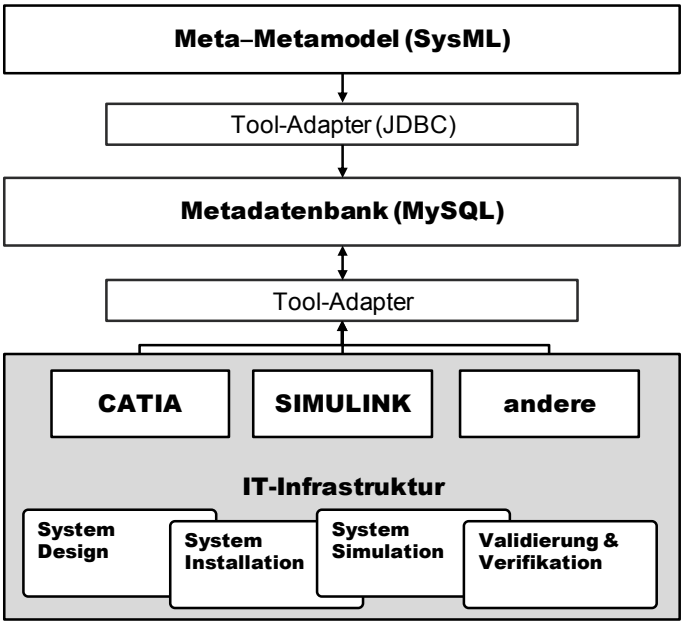


Abbildung 5-7 Schematische Infrastruktur zur modellbasierten Systementwicklung

5.4 Domänenspezifischer Entwurf

Die Grundlage der Implementierung des modellbasierten Entwurfs wird durch eine Metabeschreibung des Modells in SysML sowie einer relationalen Datenbank als Speicherort der Modelldaten gebildet.

Die vier am Entwurf von Flugzeugsystemen beteiligten Domänen, die auf das Modell zugreifen, wurden bereits in Kapitel 2.3.2 vorgestellt. Der sequentielle Arbeitsablauf zwischen den Domänen wird durch das modellbasierte Vorgehen flexibler, da einzelne Tätigkeiten nicht mehr direkt voneinander abhängen, sondern lediglich über das Metamodell verknüpft sind. Es wird im Folgenden angenommen, dass jede Domäne mit mindestens einem Anforderungsdokument (**xxRD**) und einem Partialmodell (**xxM**) arbeitet, um die domänenspezifischen Aufgaben zu erfüllen (vgl. Abbildung 5-8). Die Anforderungsdokumente und Modelle sind innerhalb der modellbasierten Umgebung jedoch mit der Systemdatenbank (**SYSDB**) verknüpft, um den zentralen Datenbestand des Systemmodells zu nutzen oder zu manipulieren.

Der vormalige Datenaustausch zwischen den einzelnen Domänen erfolgt nun ausschließlich über das Systemmodell; bestehen bleibt die Koordination von Tätigkeiten auf Basis von Kommunikation innerhalb der Domänen. Die Verwaltung der Partialmodelle mittels PDM-Systemen bleibt im Vergleich zu herkömmlichen Entwicklungsumgebungen bestehen, hinzu kommt die Verwaltung der Versionen und Releases des Systemmodells. Abbildung 5-8 stellt die Struktur eines Unternehmens und entsprechende Informations- und Datenflüsse in einer modellbasierten Umgebung dar.

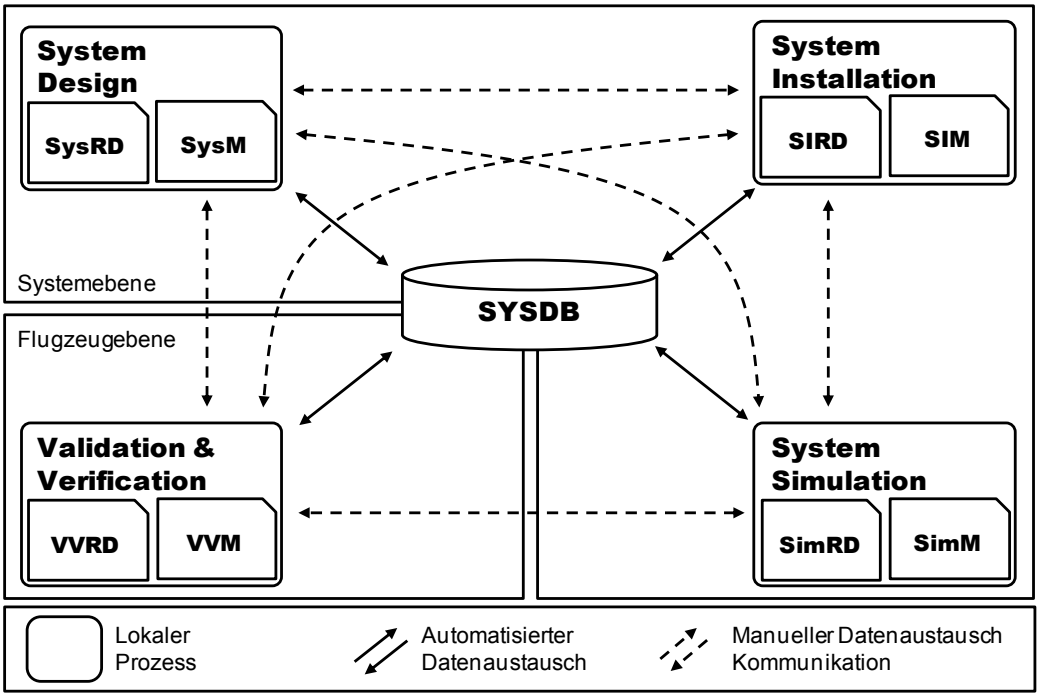


Abbildung 5-8 Domänen des Flugzeugsystementwurfs

Es ist von entscheidender Bedeutung, welche Rechnerwerkzeuge in den verschiedenen Entwicklungsdomänen eingesetzt werden und wie diese Werkzeuge mit ihrer Umgebung kommunizieren können. Wie bereits erwähnt, werden mit steigendem Spezialisierungsgrad der Tätigkeit spezifischere Werkzeuge eingesetzt. Je allgemeiner hingegen eine Tätigkeit, desto wahrscheinlicher ist der Einsatz von Werkzeugen mit breitem Anwendungsspektrum. Hierzu zählen vor allem sogenannte Office-Anwendungen. Unabhängig von der Tätigkeit gestalten sich die Dateiformate, in denen Ergebnisse des Werkzeugeinsatzes gespeichert werden. Hier lassen sich vor allem proprietäre (verschlüsselte) Datenformate und offene Datenformate unterscheiden. Offene Dateiformate lassen sich in der Regel öffnen, lesen und manipulieren, während proprietäre Datentypen dies nicht zulassen. Dafür stellt proprietäre Software oftmals gewisse Exportfunktionalitäten zur Verfügung, welche eine Interpretation der gespeicherten Daten durch andere Software zulassen. Der Export erfolgt hierbei entweder direkt in das Format des Zielsystems oder in ein offenes Datenformat. Offene Datentypen sind im CAD-Bereich vor allem *IGES* und *STEP* (SHEA, 2011), in anderen Anwendungsbereichen finden sich *CSV*-, *XML*- und Text-Dateien. Die Exportfunktionalitäten in neutrale, offene Datenformate sind allerdings meist in ihrem Umfang begrenzt. Daher können nicht alle Informationen des Ursprungsmodells für eine Interpretation durch das Zielsystem verfügbar gemacht werden. Für das Datenformat *STEP* sind verschiedene Anwendungsprotokolle (engl.: Application Protocol (AP)) in der Entwicklung, welche die Speicherung spezifischer Datensätze zulässt. Beispielsweise wird aktuell mit dem

AP242 - Managed Model Based 3D Engineering ein Format für die Automobil- und Luftfahrtindustrie definiert, welches die Speicherung von luftfahrtspezifischen Informationen ermöglicht, die über rein geometrische Informationen hinausgehen (PROSTEP, 2010). Steht keine Möglichkeit zur Verfügung, eine gewünschte Information in ein offenes Datenformat zu speichern, müssen entsprechende Schnittstellen vom Anwender programmiert werden. Das CAD-Programm CATIA stellt hierfür die externe Programmierung mittels Visual Basic und einer internen Skriptsprache zur Verfügung.

Tabelle 5-3 zeigt eine Auswahl der gängigsten Softwarewerkzeuge in der Entwicklung von Flugzeugsystemen[9]. Es ist zu erkennen, dass die meisten Werkzeuge proprietäre Datenformate verwenden und somit der Export von spezifischen Daten über ein neutrales Datenformat und entsprechend zur Verfügung gestellten Schnittstellen erfolgen muss. Für die Festlegung der auszutauschenden Daten ist der zugrunde liegende Arbeitsablauf von entscheidender Bedeutung. Der Arbeitsablauf zwischen den beteiligten Domänen kann je nach Aufgabe variieren und muss – auch im Hinblick auf eine Automatisierung – zunächst definiert werden. Es wird davon ausgegangen, dass sich alle Domänen wechselseitig beeinflussen und daher iterative Arbeitsabläufe entstehen. Sind die Arbeitsabläufe bekannt, können sie nach dem Black-Box-Prinzip abstrahiert werden und die Ein- und Ausgangsgrößen festgelegt werden. Diese Größen sind den entsprechenden domänenspezifischen Modellen zu entnehmen und in die Datenbank des Metamodells zu übertragen.

Tabelle 5-3 Dateiformate in der Entwicklung von Flugzeugsystemen

	Tool	Hersteller	Dateiformat	Verschlüsselung	Export aller Modelldaten	Export in offene Formate
Flugzeugentwurf	PrADO	TUBS/ IFL	*.dat	offen	ja	*.dat
	AAA / APP	DarCorporation	*.analysis	proprietär	ja	*.txt
	Pacelab Suite	PACE	*.plkdeoconcept, …	proprietär	nein / eingeschränkt[1]	*.step, *.xml, …
	ODIP	Airbus FPO	*.sc, …	offen	ja	
	Piano	Lissys Ltd.	*.dat, …	offen	ja	
CAD	CATIA	Dassault Systemes	*.catpart *.catproduct	proprietär	nein / eingeschränkt[2]	*.iges, *.step
	ProEngineer	PTC	*.prt, *.asm	proprietär	nein / eingeschränkt[3]	*.iges, *.step
	SolidWorks	Dassault Systemes	*.sldprt, *.sldasm	proprietär	nein / eingeschränkt[3]	*.iges, *.step
	AutoCAD	Autodesk	*.dwg	proprietär	nein / eingeschränkt[3]	*.dxf
Technical Computing	MATLAB	Mathworks	*.m, …	proprietär	ja	*.txt
	SciLab	Digiteo / INRIA	*.sc, …	proprietär	ja	*.txt
	Excel	Microsoft	*.xls	proprietär	ja	*.csv
	Mathcad	PTC	*.mcd, *.xmcd	proprietär	ja	*.xml, *.pdf, *.html
	Mathematica	Wolfram	*.wdx, *.cdf	proprietär	ja	*.xml, …
	Simulink	Mathworks	*.mdl	proprietär	nein / eingeschränkt[3]	*.xls, …
	XCOS (Scilab)	Digiteo / INRIA	*.xcos	proprietär	nein / eingeschränkt[3]	*.xml

[1] Export nur teilweise möglich, Einbindung von DLLs
[2] Export über neutrale Datenformate bzw. Parameterexport über VB & CATVBS
[3] Export über neutrale Datenformate

[9] Die vollständige Auflistung der untersuchten Softwarewerkzeuge findet sich in Annex I

5.5 Schnittstellengestaltung

Die Entwicklung von Flugzeugsystemen erfordert eine Vielzahl von Softwarewerkzeugen. Sollen Daten aus einem Werkzeug an ein anderes Werkzeug übergeben werden, liegen Datensätze oftmals in Formaten vor, in denen sie nicht weiterverarbeitet werden können. Es bedarf demnach einer Schnittstelle, welche die Daten entsprechend den Anforderungen des geplanten Austausches formatiert. Die direkte Übersetzung zwischen Metamodell und domänenspezifischen Dateiformaten scheidet aufgrund der resultierenden Vielzahl an benötigten Schnittstellen aus. Aus diesem Grund wird für die Metamodellierung eine relationale Datenbank gewählt, die einem neutralen Datenformat entspricht und die Anzahl der bereitzuhaltenden Schnittstellen reduziert (vgl. Kapitel 2.4.3).

Die Gestaltung von Schnittstellen im Bereich der Datenbankmodellierung wird unter anderem durch den ETL-Prozess (Extraktion, Transformation, Laden) beschrieben (BAUER und GÜNZEL, 2009). Darin werden Daten aus einem Quellsystem extrahiert, auf Konsistenz geprüft und angepasst sowie schließlich in die Zieldatenbank geschrieben[10]. Zur Deklaration von Tabellen und Manipulation von Daten in einer relationalen Datenbank wird die Datenbanksprache *Standard Query Language,* welche unter dem Akronym SQL bekannt ist, verwendet.

5.5.1 Extraktion der Daten

Unter dem Vorgang wird innerhalb des ETL-Prozesses die Selektion der Quelldaten verstanden, die für den Transformationsvorgang bereitgestellt werden sollen. Wie bereits mehrfach erwähnt, ist dieser Schritt aufgrund der vorhandenen Heterogenität der Ausgangsdaten verschiedenster Softwarewerkzeuge äußerst komplex und stellt die größte Herausforderung für den erfolgreichen Einsatz von Produkt- bzw. Systemmodellen dar. Die Hauptaufgabe des Extraktionsprozesses ist es also, den Zugriff auf die vielfältigen Datenformate zu ermöglichen.

Im besten Fall liegen die Quelldaten bereits in einer relationalen Datenbank vor. Dann werden die Quelldaten unter Verwendung der SQL-Sprachelemente selektiert, entsprechend der Anforderungen der Übersetzung transformiert und in die Datenbank des Metamodells überführt. Auch andere Datenbanktypen und offene Dateiformate sind zur Speicherung von Partialmodellen im Einsatz, die sich unverschlüsselt öffnen und weiterverwenden lassen. Für Quelldaten, die in Datenbanken vorliegen, existieren ver-

[10] Für die industrielle Implementierung eines ETL- Prozesses empfiehlt sich die Verwendung etablierter Transformationsstandards wie XSLT oder QVT (Query View Transformation).

schiedene Standards, welche die Extraktion von Datensätzen einheitlich gestalten sollen und bereits in vielen Werkzeugen vorhanden sind. In Annex IV wird am Beispiel der Software Matlab die Verwendung bereits vorhandener Standardschnittstellen demonstriert. Matlab stellt die Schnittstelle JDBC zur Verfügung, welche die Anbindung verschiedener Datenbanken unterstützt, unter anderem auch die für diese Arbeit benützte MySQL-Datenbank.

Im Regelfall liegen jedoch Daten von Partialmodellen in proprietären Datenformaten vor, die spezielle Übersetzer benötigen oder nur über Programmierschnittstellen zugänglich sind. Die Extraktion der Quelldaten, die je nach Anwendung sehr große Datenvolumina besitzen können, ist in diesem Fall äußerst aufwändig und ressourcenintensiv. Ein Beispiel für proprietäre Quellsysteme ist die CAD-/ PLM-Software CATIA V5, welche im Rahmen des Anwendungsbeispiels dieser Arbeit verwendet wird. CAD-Programme wie CATIA V5 sind primär für die Erstellung von 3D-Datensätzen gedacht und erst sekundär für die Erfassung der zugrundeliegenden Semantik. Dementsprechend bestehen Exportmöglichkeiten für Geometriedaten in verschiedene offene und proprietäre Datenformate während der Export von abgelegtem Ingenieurwissen oder Analysedaten nur in Grundzügen vorgesehen ist. Es besteht zum Beispiel die Möglichkeit, Parameter in unterschiedliche Dateiformate (z.B.: *.xls, *.csv) zu exportieren. Die externe Speicherung von Messungen (z.B.: Masse, Länge, Volumen) wird allerdings nur über benutzerdefinierte Skripte ermöglicht. In Annex V ist die Extraktion von Parametern und Messungen mittels der Programmierschnittstelle von CATIA V5 detailliert erläutert.

5.5.2 Transformation der Daten

Innerhalb dieses Teilprozesses werden die aus einem Partialmodell stammenden Quelldaten in ein einheitliches Datenbankschema transformiert. Die Transformation beschreibt dabei die Anpassung und Erweiterung der Daten an die vom Metamodell vorgeschriebenen Zielstrukturen. Zunächst werden Datensätze verbessert oder korrigiert, um die einheitliche Verwendung von Formaten zu gewährleisten und dem Syntax des Zielsystems zu entsprechen (Syntaktische Transformation). Weiterhin werden die Datensätze um Duplikate bereinigt und um Sprachelemente der SQL angereichert (Semantische Transformation), um den folgenden Teilprozess *Laden* ausführen zu können (BAUER und GÜNZEL, 2009).

Die Sprachelemente der SQL lassen sich in drei Klassen aufteilen (VOSSEN, 2008):

- Data Definition Language (DDL): Befehle zur Definition von Tabellen und Datenbankstrukturen (Bsp.: `create table, drop table`)
- Data Manipulation Language (DML): Befehle zur Datenmanipulation und Datenabfrage (Bsp.: `select, insert, update`)
- Data Control Language (DCL): Befehle zur Kontrolle von Zugriffsrechten (Bsp.: `grant, revoke`)

Transformationsskripts bedienen sich hauptsächlich Elementen der DML, um die Dateneinträge in die Struktur des Metamodells vorzubereiten. (Anmerkung: die Struktur des Metamodells wird aus dem SysML-Modell mit Hilfe der Standardschnittstelle JDBC unter Verwendung von DDL- und DML-Elementen automatisiert erzeugt.)

5.5.3 Laden der Daten

Nachdem die Quelldaten aus einem Partialmodell erfolgreich extrahiert und in ein SQL-Skript transformiert worden sind, können sie durch das DBMS geladen werden. Danach steht der geladene Datensatz allen am Entwurfsprozess beteiligten Domänen und Werkzeugen zur Verfügung. Die Daten können mittels der allen Prozessbeteiligten bekannten Metamodellstruktur ausgelesen werden und in der eigenen Modellierungsumgebung weiterverwendet werden.

Es bleibt festzuhalten, dass die begrenzten Exportfunktionalitäten einer Vielzahl von Werkzeugen der ganzheitlichen Integration von Arbeitsabläufen und somit die modellbasierte Vorgehensweise im Wege stehen. Deshalb bedarf es vor der Implementierung eines modellbasierten Entwicklungsablaufes einer genauen Analyse der Arbeitsabläufe und der jeweiligen Ein- und Ausgangsgrößen. Diese müssen wiederum von den eingesetzten Werkzeugen erzeugt und in der zentralen Systemdatenbank gespeichert werden können. Ist die Exportfunktionalität eines Werkzeuges nicht vorhanden, müssen die Daten von Hand erzeugt werden und ein Bruch in der (automatisierten) Werkzeugkette wäre die Folge.

6 EXEMPLARISCHE ANWENDUNG FLUGZEUGSYSTEME

Schritt für Schritt wurde in den vorangegangenen Kapiteln die Notwendigkeit des modellbasierten Vorgehens in der Systementwicklung erläutert und die Grundlagen für eine praktische Umsetzung der Theorie geschaffen. Im folgenden Kapitel soll am Beispiel des Entwurfs einer Flugzeugklimaanlage (engl.: Environmental Control System (ECS)) das Potenzial der Vorgehensweise dargestellt werden.

Das Anwendungsbeispiel Flugzeugklimaanlage wurde gewählt, da das ECS ein repräsentatives, mechatronisches System ist, welches von unterschiedlichen Entwicklungsdomänen entworfen und getestet wird. Durch die sicherheitskritische Funktion ist das System einer Vielzahl von Regularien und Anforderungen unterworfen und verlangt dementsprechend kontinuierliche Verifikation der Entwicklungstätigkeiten und Validierung der Systemfunktion. Aufgrund der geometrischen Dimensionen des Systems ist die Interaktion mit anderen Systemen besonders hoch und verlangt vielfältige DMU-Studien im iterativen Entwurfsablauf. Die Eingliederung des ECS in die Gesamtarchitektur der Leistungserzeuger und -verbraucher in einem typischen Verkehrsflugzeug wird in Abbildung 6-1 dargestellt.

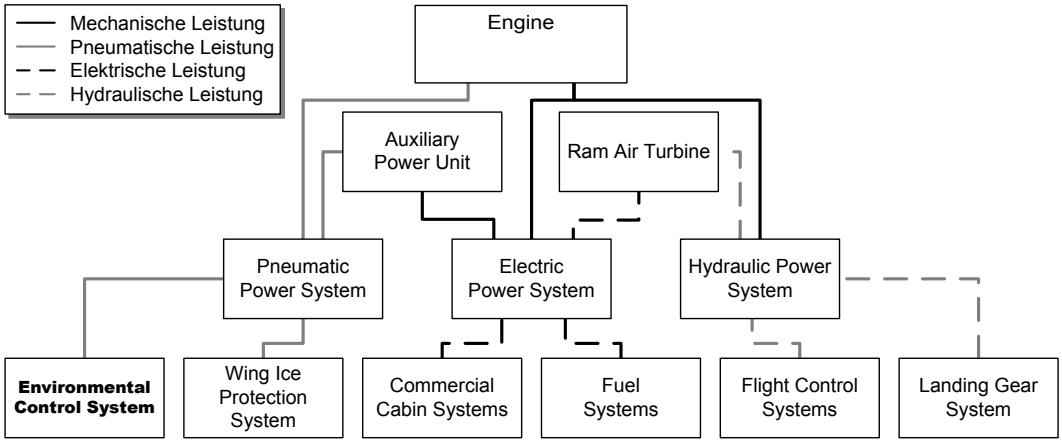


Abbildung 6-1 Einordnung der ECS in die Flugzeugarchitektur
Quelle: LISCOUET-HANKE, 2008; leicht modifiziert

Die Fallstudie definiert zunächst die funktionalen Anforderungen an das ECS und beschreibt danach die am Entwurf beteiligten Domänen derart, dass die domänenspezifi-

schen Anforderungen an das Metamodell definiert werden können. Darauf aufbauend erfolgt gemäß dem Prinzip der Segmentierung die Festlegung relevanter Systemsichten. Die verschiedenen Sichten orientieren sich dabei an der Struktur der beteiligten Entwicklungsdomänen und dem Verständnis der Entwickler, den eingesetzten Werkzeugen und den über Domänengrenzen hinweg entstehenden Datenfluss. Anschließend wird das Metamodell anhand der definierten Systemsichten um Elemente und deren Relationen erweitert. Um gleichartige Elemente nicht immer neu definieren zu müssen, werden nach den Prinzipien objektorientierter Programmierung Typen definiert, die schließlich um Attribute erweitert werden, um eine exakte Systembeschreibung zu ermöglichen. Diese Tätigkeiten erfolgen mit SysML als Modellierungssprache und erzeugen eine Datenbankstruktur, die als Basis für die Instanziierung des Systemmodells dient. Schließlich wird ein Ausschnitt der Systementwicklung am Beispiel ECS erläutert und das Potenzial der modellbasierten Vorgehensweise demonstriert.

6.1 Systemanforderungen und –funktionen

Die Klimaanlage von Verkehrsflugzeugen mit Druckkabine hat die Aufgabe, ein für den Passagier und die Besatzung lebensfreundliches Klima herzustellen. Darunter fällt unter anderem die Regelung der Temperatur, des Drucks und der Luftfeuchtigkeit in den individuell geregelten Kontrollvolumina des Flugzeuges: Cockpit, Passagierkabine und Frachtraum. Die meisten am Markt befindlichen Passagierflugzeuge verwenden Zapfluft aus den Triebwerken, die entsprechend der Vorgaben für die Kabinenluft konditioniert wird. Die Regelung der Temperatur erfolgt über Ventile, die heiße Zapfluft mit bereits gekühlter Luft aus der Klimaanlage vermischen. Der gewünschte Kabinendruck wird über kontrolliertes Ablassen verbrauchter Luft durch ein Überdruckventil am Heck des Flugzeuges erreicht.

Die beschriebene Architektur einer Flugzeugklimaanlage basierend auf Zapfluft ist heute in den meisten Verkehrsflugzeugen zu finden und bildet die Grundlage dieser Fallstudie. Der Hersteller Airbus wendet die Zapfluftarchitektur auch zukünftig in dem Modell A350 an. Da der Anteil der Klimaanlage am Gesamttreibstoffverbrauch allerdings etwa 5% beträgt und hiervon über 80% auf die Zapfluftentnahme fallen (LEHLE, 2006), werden aktuell alternative Architekturen untersucht. Der Hersteller Boeing hat für das Modell 787 eine komplett elektrische Klimaanlage eingeführt, die durch zusätzliche Generatoren im Triebwerk elektrische Stauluftkompressoren antreiben, was den Anteil am Treibstoffverbrauch reduzieren soll. Welche Architektur sich in Zukunft durchsetzen wird, bleibt jedoch offen. Das EU geförderte Projekt MOET bestätigte in diesem Zusammenhang eine Gewichtszunahme von MEA-Flugzeugarchitekturen

(MOET, 2009), wohingegen BUTTERWORTH-HAYES (2009) mehrere Quellen zitiert, die eine Gewichtsreduktion einer MEA-Gesamtarchitektur propagieren.

Die Grenzbereiche der Zustandsgrößen von Kabinenluft sind in verschiedenen Richtlinien festgehalten:

Belüftung:

- Minimale Belüftung: 4,7l/s für die Crew (JAR-25 § 831(a)). Üblicherweise werden aber pro Person mindestens 7,8l/s zugeführt (DAVIES, 2003).
- Im Falle des Versagens einer Komponente muss die Frischluftversorgung mit mindestens 3,1l/s pro Person ohne den Anteil rezirkulierter Luft aufrecht erhalten werden (JAR-25 § 831(c)).
- Maximale Luftgeschwindigkeit in Passagiernähe von 0,2m/s um Luftzug zu vermeiden (AIR 1168/3).

Temperaturkontrolle (ARP85E, 1991):

- Normaler Einstellbereich der ECS: 18°C – 30°C.
- Kühlen von 46°C auf 27°C in 30min[11]
- Aufheizen von -32°C auf 21°C in 30min[1]

Druckkontrolle:

- Maximale Kabinendruckhöhe 8000ft[12] (JAR-25 §841(a))
- Maximale Änderung des Kabinendrucks entspricht einer Steigrate von 2,5m/s (500ft/min) bzw. einer Sinkrate von 1,5m/s (300ft/min) (SAE, 2010).

Um aus den hier genannten behördlichen Anforderungen eine einheitliche Formulierung für die Systemfunktion nach den Anforderungen der Funktionalen Produktentwicklung (vgl. Kapitel 3.1.2) abzuleiten, ist eine Umformulierung notwendig. Dazu kann die Vorgehensweise nach MEIER (2005) verwendet werden, die eine Funktion stets als Kombination aus Substantiv und Verb (Infinitiv) formuliert und eine Präzisierung durch Attribute und Adjektive zulässt. Für die rechnertechnische Interpretation von Systemfunktionen ist zudem die Definition von Richtwerten bzw. Ober- und Untergrenzen vorzunehmen. Von entscheidender Bedeutung ist die einheitliche Definition der Systemfunktionen, um die Weiterverwendbarkeit im nachfolgenden Entwicklungsablauf zu unterstützen.

[11] Ohne Passagiere und weitere Wärmequellen sowie geschlossenen Türen
[12] Bei ECS- Versagen nicht mehr als 15000ft

6.2 Domänen und Arbeitsabläufe

Die am Entwurf beteiligten Domänen und ihre Aufgabenbereiche wurden bereits in Kapitel 2.3.2 erläutert. Am Beispiel der ECS-Entwicklung soll nun der Einfluss der modellbasierten Infrastruktur auf die Arbeitsabläufe der Systementwicklung dargestellt werden. Abbildung 6-2 (oben) zeigt zum einen den herkömmlichen Ablauf der Entwicklungstätigkeiten, der durch die sequentielle Bearbeitung von Partialmodellen gekennzeichnet ist. Ebenso sind iterative Arbeitsschritte zwischen zwei Domänen dargestellt, die beispielsweise während einer Fehlerbehebung auftreten und den Datenfluss an nachfolgende Domänen verzögern. Zum anderen ist der modellbasierte Entwurfszyklus dargestellt (Abb. 6-2 unten), welcher eine stärkere Parallelisierung der Entwurfstätigkeiten zulässt und die Entwurfstätigkeiten voneinander entkoppelt. Die Domänen sind lediglich über das in der Systemdatenbank gespeicherte Metamodell miteinander verknüpft, wodurch unabhängiges Arbeiten ermöglicht wird. In Abhängigkeit des Tätigkeitsprofiles und des Automatisierungsgrades domänenspezifischer Tätigkeiten steigt oder sinkt der Grad der Parallelisierung von Domänen, was in Abbildung 6-2 durch gestrichelte Pfeile dargestellt ist. Dadurch entsteht eine potenzielle Reduktion der Entwicklungszeit, die je nach Anwendungsfall unterschiedlich ausfallen kann.

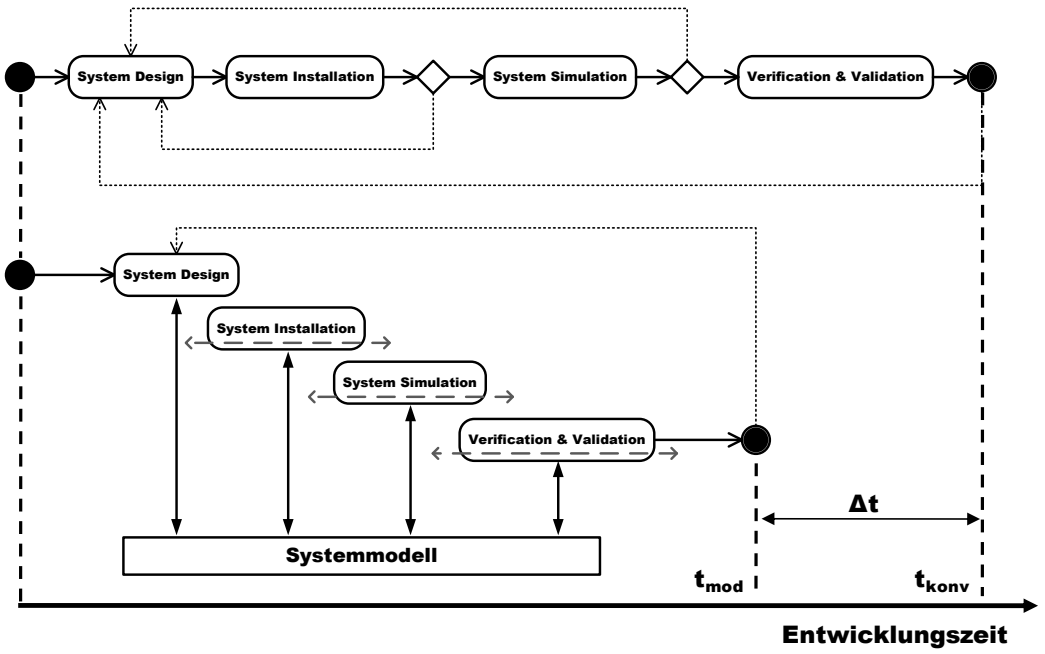


Abbildung 6-2 Herkömmlicher (oben) und modellbasierter Entwurfszyklus (unten)

Wie die Entkopplung der Domänen konkret erfolgen kann, wird in den folgenden Abschnitten am Beispiel der ECS-Entwicklung erläutert. Dabei stehen der Entwurf des Metamodells mit SysML und der Datenbankstruktur sowie die Integration in eine Softwarelandschaft mit bestehenden Arbeitsabläufen im Vordergrund. Zum besseren

Verständnis fasst Tabelle 6-1 die Aufgabenbereiche und eingesetzten Werkzeuge der Entwicklungsdomänen sowie den resultierenden Informationsaustauschbedarf zusammen.

Tabelle 6-1 Domänen und Aufgaben in der ECS Entwicklung

Domäne	Aufgaben	Werkzeuge	Daten
System Design	Synthese, Function-to-Form Mapping	Doors, Excel, Matlab	Input: Anforderungen Output: Funktionen, Systemparameter, Metamodellstruktur
System Installation	DMU Integration und Analyse	CATIA V5	Input: Systemparameter Output: 3D- Modell, 3D- Analysedaten
System Simulation	Simulation der Systemleistung	Matlab / Simulink	Input: Systemparameter, 3D- Analysedaten Output: Systemverhalten
Verfikation & Validierung	Verifikation der Systemleistung	Matlab	Input: Systemverhalten, Funktionen Output: Systembewertung, Änderungsanweisungen

(Anmerkung: Nachdem für die Erstellung des Anwendungsbeispiels keine tatsächlichen Entwicklungsdomänen mit verschiedenen Bearbeitern existieren, werden die numerischen Aktivitäten der vier fiktiven Domänen als eigenständige Skripte in physikalisch getrennten Speicherorten abgelegt. Die so entstehende Entwicklungsumgebung entspricht einem räumlich getrennten Entwicklungsteam, deren Mitglieder unterschiedlichen Domänen angehören. Die Infrastruktur dieser Entwicklungsumgebung ist in Annex III dargestellt.)

6.2.1 System Design

6.2.1.1 Vorbereitende Tätigkeiten

Die primäre Aufgabe der Systemingenieure ist die Strukturierung der Anforderungen an ein System, die Definition der Systemfunktionen und die Ableitung einer Systemstruktur, die den Anforderungen entspricht. Dazu ist zunächst eine Erfassung und Kategorisierung der Anforderungen notwendig. In SysML steht hierzu das Anforderungsdiagramm zur Verfügung. In diesem Diagrammtyp werden textbasierte Anforderungen definiert und Beziehungen zwischen weiteren Anforderungen, Systemelementen und Anwendungsfällen graphisch dargestellt, um die Verfolgbarkeit einer Anforderung in der Systemstruktur zu gewährleisten. Abbildung 6-3 zeigt beispielhaft ein interaktives SysML-Anforderungsdiagramm in dem die Hierarchie und die Herkunft der Anforderungen sowie die beeinflussten Strukturelemente dargestellt sind. Allen Verknüpfungen zwischen den Elementen wird ein Stereotyp zugewiesen, der den Typ der Verknüpfung genauer spezifiziert. So sind beispielsweise der Block *ECS* und die Anforderung *provideOperatingEnvironment* über eine Erfüllungsbeziehung (*satisfy*) verbunden, während die Anforderung *provideAdequatePressure* über eine Ableitungsbe-

ziehung (*derive*) mit einem Hyperlink zu einem externen *JAR-25$841a*-Dokument verknüpft ist. (Anmerkung: Es ist zu erkennen, dass die graphische Diagrammdarstellung bei einer Vielzahl von Elementen schnell an ihre Grenzen gerät. Der Wechsel zu einer baumartigen Darstellung wird in diesem Fall empfohlen (NOMAGIC, 2012)).

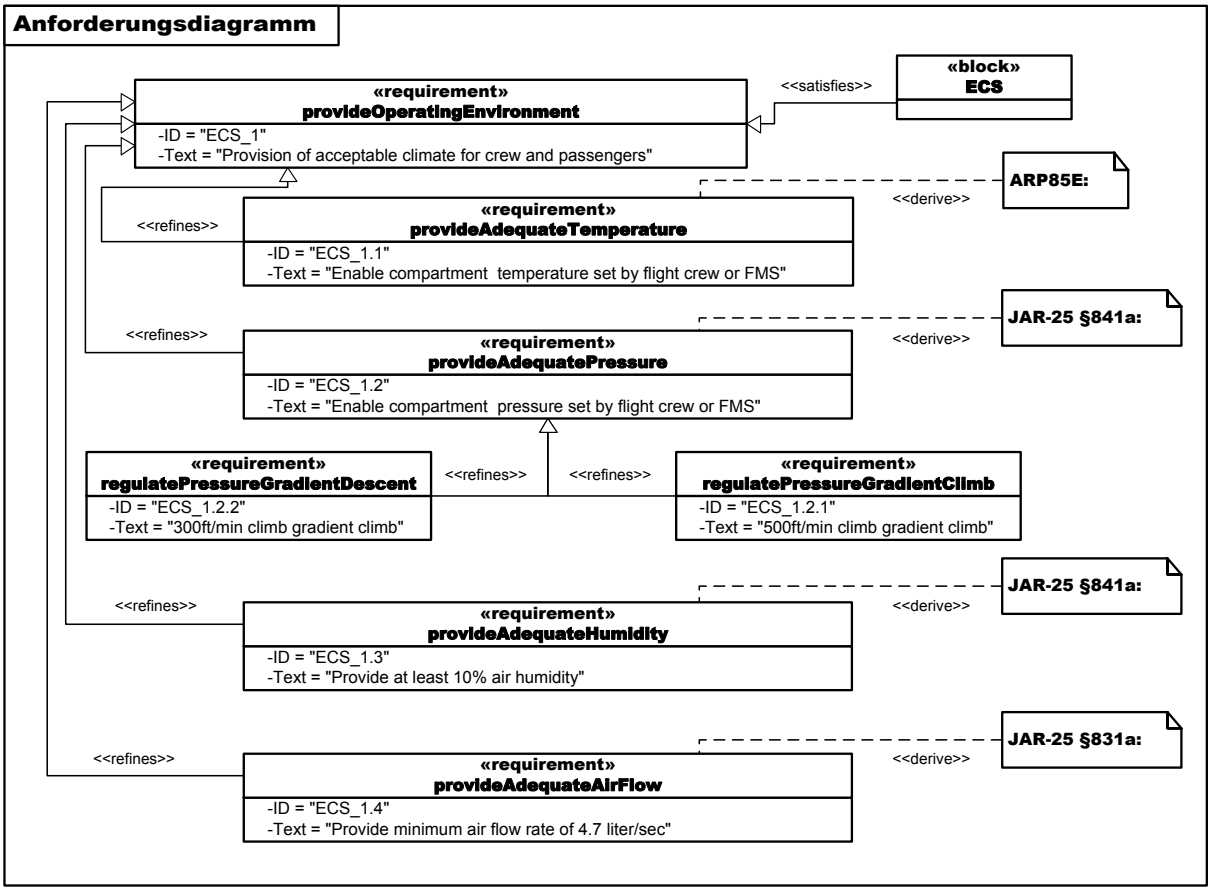


Abbildung 6-3 Interaktives SysML Anforderungsdiagramm

Die tiefe Integration von Anforderungen in die Synthesephase bildet die Grundlage für robuste Systemarchitekturen (vgl. Kapitel 3.1.2) wobei die Anwendbarkeit durch die graphisch orientierte Notation der SysML unterstützt wird. Alle Elemente, die in SysML-Diagrammen dargestellt werden, können als Hyperlinks verwendet werden, um zwischen verschiedenen Diagrammtypen zu wechseln. Beispielsweise können in einem Anforderungsdiagramm Hyperlinks zu Notizelementen, die den Ursprung der Anforderung beschreiben, oder zu Elementen innerhalb eines Blockdiagramms hergestellt werden. Dem Anwender wird so die Möglichkeit geboten, interaktiv in dem SysML-Modell zu navigieren. Eine bessere Nachvollziehbarkeit (engl.: traceability) von Entwurfsentscheidungen wird auf diese Weise unterstützt.

Sind alle Anforderungen bekannt, erfasst und in Relation zueinander gesetzt, wird durch die Definition von Anwendungsfällen das geforderte Systemverhalten genauer spezifiziert. Durch die Festlegung der relevanten Anwendungsfälle unter Einbeziehung

von Prozessbeteiligten aller Entwicklungsabteilungen wird die ganzheitliche Sicht auf ein System verbessert und gleichzeitig die Fehlerwahrscheinlichkeit in späteren Phasen reduziert. Für die Definition von Anwendungsfällen von luftfahrttechnischen Systemen, sind die nach ATA-Kapiteln geordneten ARP-Richtlinien heranzuziehen (vgl. Kapitel 3.2.3). Weiterhin dienen die Anwendungsfälle und die entsprechende Dokumentation als Vorlage für nachfolgende Tätigkeiten wie der *Systemsimulation* oder *Verifikation und Validierung*.

Tabelle 6-2 zeigt die Anwendungsfälle eines ECS, die für den Entwurf innerhalb dieser Arbeit berücksichtigt werden.

Tabelle 6-2 Definition von Anwendungsfällen des ECS

	Anwendungsfall			**Beschreibung**
Druckregelung	UC_1.1	Flugbetrieb	Steigflug	Maximaler Druckgradient
	UC_1.2	Flugbetrieb	Reiseflug	Minimaler Kabineninnendruck
	UC_1.3	Flugbetrieb	Sinkflug	Maximaler Druckgradient
	UC_2.1	Notfallbetrieb	Druckverlust	Leckage im bedruckten Bereich
	UC_2.2	Notfallbetrieb	Zapfluftausfall	Versagen der Zapfluftversorgung
	UC_2.x	…	…	…
Temperaturregelung	UC_3.1	Standfall	Heißer Tag	Abkühlen des Kontrollvolumens
	UC_3.2	Standfall	Kalter Tag	Aufheizen des Kontrollvolumens
	UC_3.x	…	…	…

6.2.1.2 Erstellung des Metamodells

Mit Hilfe der erfolgten Anforderungsdefinition und der Erstellung von Anwendungsfällen kann das *System Design* die Systemarchitektur bestimmen. Diese kann von Grund auf neu definiert werden oder von bereits vorhandenen Strukturen aus Vorgängermodellen abgeleitet werden. Dazu sind zunächst die Festlegung der Systemelemente und deren Eigenschaften notwendig. Danach erfolgen die Definition der Struktur der Systemelemente und ihrer internen Verbindungen sowie die Definition der Schnittstellen zur Systemumwelt. Das Ergebnis dieser Tätigkeiten wird in einem SysML-Blockdiagramm festgehalten.

Die Struktur der Elemente des Blockdiagramms stellt die generische Systembeschreibung des ECS-Systems dar. Dabei werden die Elemente durch Attribute beschrieben und der Datentyp der Attribute definiert. Diese Tätigkeit muss in Übereinstimmung mit allen Domänenvertretern erfolgen, denn die hier festgelegten Datenformate müssen später von allen beteiligten Werkzeugen einheitlich verwendet werden können. Auf Basis des Blockdiagramms wird mittels der JDBC-Schnittstelle automatisch der SQL-Code generiert, der zur Erzeugung der Datenbankstruktur dient. Nach dem Einle-

sen des SQL-Code durch das DBMS steht die Datenbank zur Instanziierung des Systemmodells und dem Zugriff der Domänenschnittstellen zur Verfügung.

Üblicherweise entstehen im Laufe der Entwicklung mehrere Varianten oder Versionen einer Systemstruktur. Diese Änderungen umfassen demnach die Erweiterung bzw. die Reduzierung des Systemmodells um Elemente und Relationen. Wird eine Änderung des Systemmodells in SysML vorgenommen, kann der entsprechende SQL-Code zur Erzeugung/Löschung von Tabellen zur Adaption einer vorhandenen Systemdatenbank generiert werden. In der industriellen Praxis wird jedoch die Erzeugung einer neuen Datenbank im Rahmen eines PDM-Systems erfolgen. SQL-Datenbanken lassen analog zu anderen Produktdatensätzen eine Versionierung zu und ermöglichen auf diese Weise Vergleichsstudien alternativer Systemarchitekturen sowie die Dokumentation des Produktentwicklungsprozesses.

6.2.2 System Installation

Das Fachgebiet *System Installation* hat zur Aufgabe, verschiedene Flugzeugsysteme in den begrenzten Bauräumen eines Flugzeugrumpfes in optimaler Weise zu verteilen. Diese Aufgabe wird von einer Vielzahl von Aspekten beeinflusst. Dazu zählen der Einbau von räumlich nicht trennbaren Systemen (engl.: packages), Unverträglichkeiten von Systemen (z.B.: EMV) oder die Minimierung des Kabelgewichts. Um ein optimales Ergebnis dieser Einbauuntersuchungen (engl.: space allocation) zu erhalten, werden diese Tätigkeiten heute durch eine Vielzahl von Algorithmen unterstützt. Beispielsweise wird durch *BinPacking*-Algorithmen eine Anzahl von Boxen unter verschiedenen Randbedingungen in vorhandene Bauräume „gepackt“. Ein weiteres Beispiel ist die automatische Verlegung von Kabelrouten oder Luftleitungen in einem Flugzeugrumpf unter Berücksichtigung von Inkompatibilitäten, Biegeradien und Mindestabständen. Grundlage für alle Installationsaufgaben ist ein DMU, welches die Flugzeugsysteme aller ATA-Kapitel integriert und 3D-Analysemöglichkeiten bietet. Für die Entwicklung einer ECS-Architektur sind alle Komponenten in entsprechenden Bauräumen zu integrieren, um die Interaktion mit anderen Systemen überprüfen und Messungen von Systemparametern vornehmen zu können.

Für die exemplarische Anwendung dieser Arbeit wird auf Basis einer Flugzeuggeometrie in der Größe des Airbus A320 das Modell der Luftleitungen einer Zapfluftklimaanlage generiert. Das digitale Modell wird beispielsweise für die Demonstration der automatisierten, bidirektionalen Kommunikationsfähigkeit zwischen CATIA V5 und der Systemdatenbank genutzt. Weitere Anwendungsszenarien sind die automatisierte Geometrieerzeugung, Kollisionsanalyse und Fehlerberichterstellung. Die genannten

Analysen auf Basis von CATIA V5 zeigen die Integrationsfähigkeit von Softwarewerkzeugen mit proprietären Datenformaten in eine integrierte Entwicklungsumgebung. Die Anwendbarkeit des modellbasierten Vorgehens für die Entwicklung von Flugzeugsystemen, bei der eine Vielzahl dieser Werkzeuge eingesetzt wird (vgl. Tabelle 5-3), wird durch diesen Umstand deutlich gesteigert. Abbildung 6-4 zeigt Elemente der Primärstruktur und der ECS-Luftleitungen (rechts) als einen beispielhaften Ausschnitt des DMU.

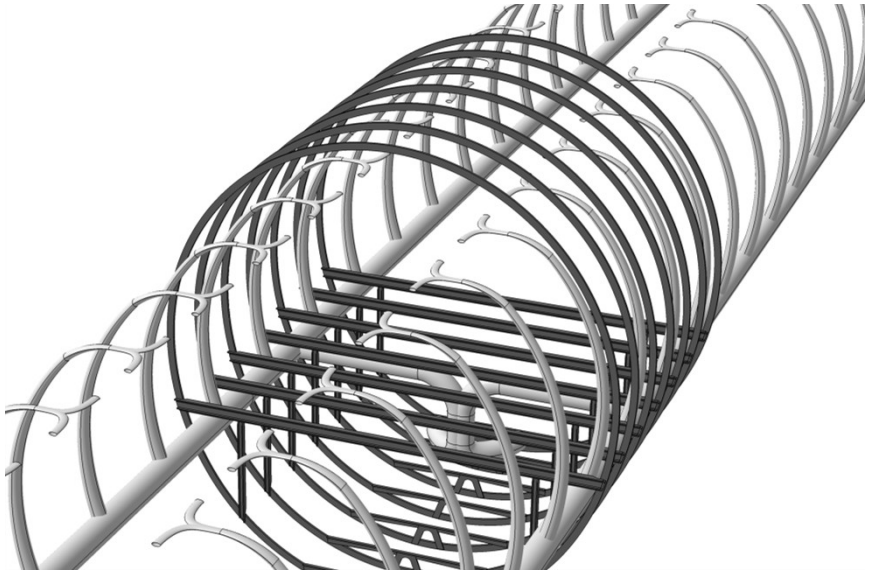


Abbildung 6-4 Ausschnitt des Digital Mockup der Domäne System Installation

6.2.3 System Simulation

Um die Funktionsweise eines Systems simulieren zu können, muss zunächst eine vereinfachte Beschreibung eines geeigneten Ausschnittes des Betrachtungsgegenstandes erfolgen, welche die Grundlage der anschließenden Berechnungen bildet (BUNGARTZ, 2009). Für die Simulation des ECS sind demnach alle Elemente des Systems zu identifizieren und das entsprechende physikalische Verhalten abstrahiert zu beschreiben. In den folgenden Abschnitten wird die Modellbildung der ECS-Komponenten erläutert, um den domänenübergreifenden Informationsfluss, welcher die Struktur des Metamodells bedingt, darzustellen.

6.2.3.1 Triebwerk

Die Hauptverdichtung der Luft, die für den Kühlprozess der Klimaanlage notwendig ist, wird von den Verdichterstufen des Triebwerks übernommen. Da der Leistungsbedarf der Klimaanlage zeitlich nicht dem Profil der Triebwerksleistung entspricht, sind meist zwei Zapfpunkte an verschiedenen Verdichterstufen vorhanden. Im Steig- und Reiseflug wird Luft aus einer niedrigen Verdichterstufe verwendet, im Sinkflug wird

aufgrund der niedrigen Triebwerksleistung Luft aus einer höheren Verdichterstufe entnommen. Abbildung 6-5 zeigt den beispielhaften Verlauf der Flughöhe und den entsprechenden Schubverlauf einer typischen Transportmission (vgl. EGELHOFER, 2009), die für dieses Anwendungsbeispiel auf Basis eines BADA[13]-Punktmassenmodells berechnet wurde.

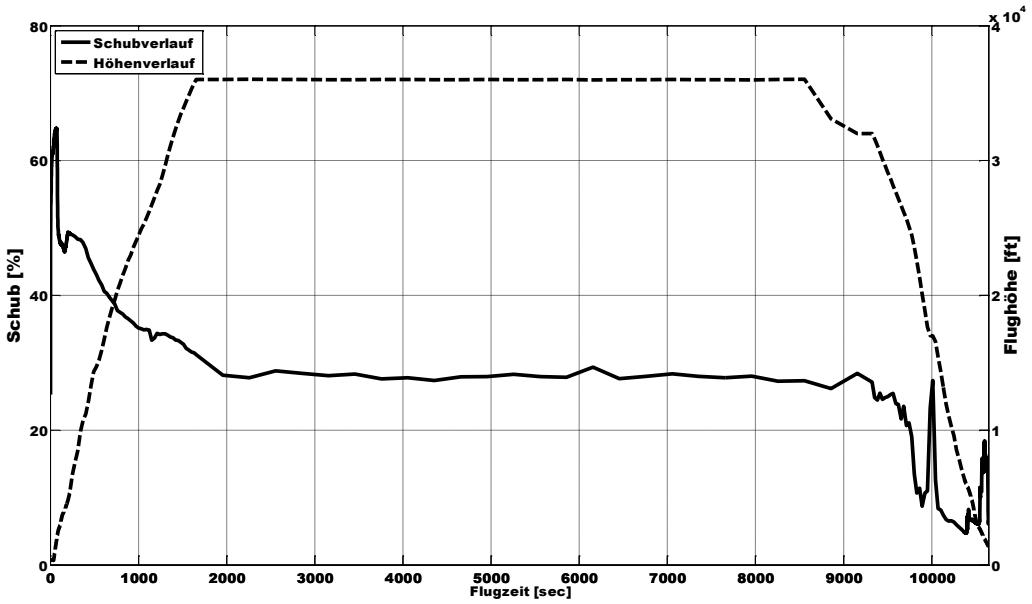


Abbildung 6-5 Schub- und Höhenverlauf einer A320 Mission

Um aus einem realen oder durch Flugleistungssimulation berechneten Schubverlauf des Triebwerks die Zustandsgrößen der Zapfluft berechnen zu können, wird der schubabhängige Druckanstieg der gewählten Verdichterstufen berechnet.

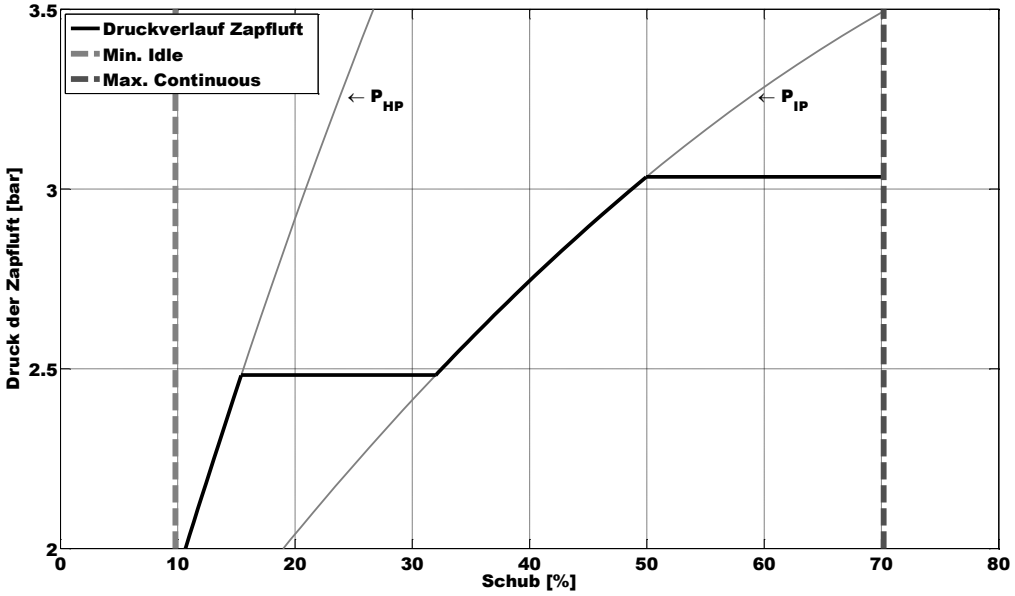


Abbildung 6-6 Schubabhängiger Druck der Zapfluft

[13] BADA: Base of Aircraft Data (http://www.eurocontrol.int/eec/public/standard_page/proj_BADA.html)

Die daraus entstehende Charakteristik des Druckes der für die Klimatisierung notwendigen Zapfluft wird in Abbildung 6-6 dargestellt. Das Triebwerksmodell der ECS-Simulation besteht demnach aus einem Schubmodell und den Verdichterkennfeldern der entsprechenden Zapfpunkte.

6.2.3.2 Klimaanlage

Nach den eben beschriebenen Prinzipien wird die Umgebungsluft im Verdichter des Triebwerks komprimiert und tritt als Zapfluft im Reiseflug mit etwa 2,5bar und 200°C in die Klimaanlage (engl.: air cycle machine (ACM)) ein, wo durch einen linksläufigen Joule-Prozess klimatisierte Luft erzeugt wird. Zunächst durchläuft die Zapfluft den ersten Wärmetauscher, der von Stauluft (engl.: ram air) gekühlt wird. Die Wärmetauscher von Flugzeugklimaanlagen sind im Gegensatz zu stationären Anlagen luftgestützt und verwenden keine Kühlflüssigkeiten (LEHLE, 2006). Um eine hohe Prozesseffizienz zu erreichen, wird nach dem ersten Wärmetauscher erneut verdichtet und ein zweiter Wärmetauscher durchlaufen. In der Regel kommen Kreuzstromwärmetauscher (1. WT) und Kreuzgegenstromwärmetauscher (2. WT) aus Aluminium zum Einsatz (LEHLE, 2006). Danach durchläuft die Luft eine Turbine, in der sie expandiert und abkühlt. Die entstehende Rotationsenergie treibt den Kompressor und das Gebläse der Wärmetauscher an. Die Temperatur am Ausgang der Turbine beträgt ungefähr 5°C und wird in der Mischkammer wiederum mit heißer Zapfluft vermischt, um das gewählte Temperaturniveau der Kabine zu erhalten. Der ideale und reale Verlauf der Zustandsgrößen Temperatur und Entropie werden in Abbildung 6-7 dargestellt (Anmerkung: die Zustandsänderungen des Wasserabscheiders sind nicht dargestellt).

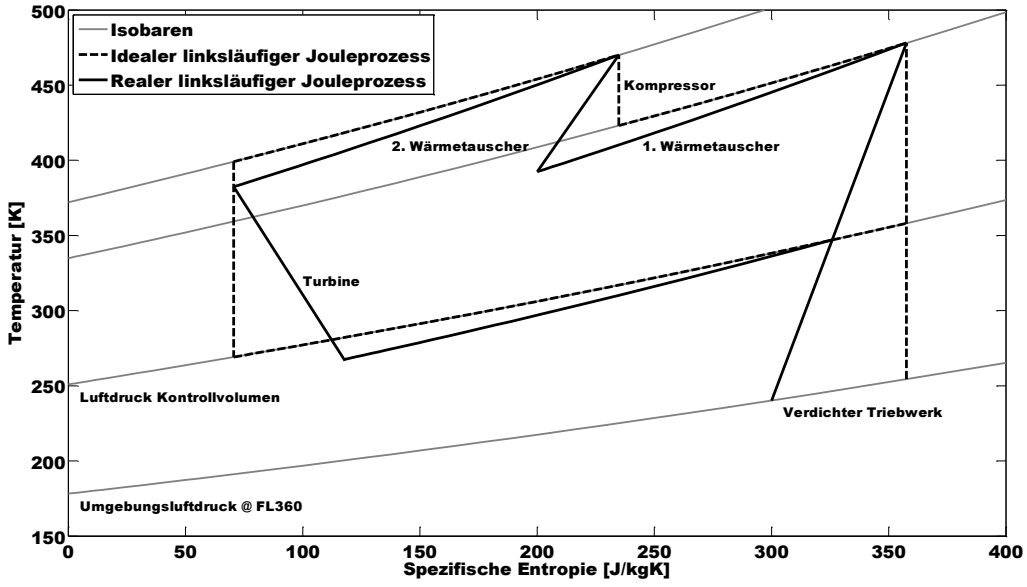


Abbildung 6-7 T-s-Diagramm der ACM (@ FL360)

Für die Systemsimulation ist die Modellierung der physikalischen Prinzipien der Kältemaschine notwendig, um die Zustandsgrößen der Kabinenluft und den hierfür benötigten Energieaufwand berechnen zu können. Aus dem Energiebedarf der ACM werden die geometrischen Dimensionen der Komponenten berechnet, welche nun zum Beispiel für Integrationsanalysen innerhalb des DMU zur Verfügung stehen.

6.2.3.3 Klimazonen

Die primäre Regelgröße der ACM stellt die Temperatur der verschiedenen Klimazonen dar. Jede Zone stellt ein Volumen definierter Größe dar, welches mit dem Medium Luft gefüllt wird. Die Änderung der Temperatur innerhalb des Kontrollvolumens kann durch die Annahme isobarer Zustandsänderungen berechnet werden (KWIATKOWSKI, 2006). Die Enthalpie setzt sich dabei aus innerer Energie und Volumenarbeit zusammen. Die Änderung der inneren Energie resultiert aus den vorhandenen Wärmequellen und Wärmesenken. Beispiele für Wärmequellen sind Passagiere, elektronische Systeme oder die Strahlungswärme der Sonne. Der Wärmeübergang durch die Begrenzungsflächen des Kontrollvolumens stellt hingegen eine Wärmesenke dar. Die hinreichend genaue Modellierung der Klimazonen eines Flugzeuges benötigt eine Vielzahl an geometrischen und physikalischen Größen. Hierzu zählen das Volumen der Klimazonen, Anzahl Passagiere, Frameabstand, Framebreite, Fläche der Fenster oder die Wärmeleistung des elektrischen Equipments.

6.2.3.4 Mischkammer

Innerhalb einer ECS werden mehrfach Luftströme mit verschiedenen thermodynamischen Zuständen vermischt. Dazu gehört die Vermischung von klimatisierter Luft aus der ACM mit heißer Zapfluft oder mit rezirkulierter Kabinenluft, welche zur Nutzung der enthaltenen Enthalpie und Luftfeuchte erneut beigemischt wird. Für die Modellbildung wird angenommen, dass Luft ein ideales Gas ist und die Masse der Luft in der Mischkammer ungeachtet der Druckänderungen konstant bleibt.

6.2.3.5 Leitungselemente

Verschiedene Luftleitungen sind notwendig um die Luft von den Zapfpunkten im Triebwerk über die Klimaanlage und Mischkammer zu den Klimazonen zu leiten. Diese Leitungen unterscheiden sich aufgrund der unterschiedlichen Zustandsgrößen der Luft in ihrer Geometrie und den verwendeten Materialien. Entscheidendes Merkmal für die Dimensionierung einer ECS ist der Druckabfall in den Rohrleitungen, welcher durch die innere Reibung, den Rohrverlauf und den Massendurchsatz bestimmt wird. Für die ECS-Simulation werden nur Hauptleitungen modelliert, bei denen der Druck

am Leitungseinlass und der benötigte Druck am Leitungsende bekannt sind. Auf Basis der Leitungsparameter kann der benötigte Luftmassenstrom iterativ bestimmt werden. Die Änderung der Zustandsgrößen in der Vielzahl von Verteilungsleitungen wird aufgrund des hohen Modellierungsaufwandes als vernachlässigbar betrachtet und im Rahmen dieser Arbeit lediglich über einen Verlustfaktor berücksichtigt (vgl. LISCOUET-HANKE, 2008).

6.2.3.6 Ventile

Gängige Ventile in pneumatischen Maschinen weisen entweder zwei bis drei diskrete und durch Elektromagnete gesteuerte Öffnungsstellungen auf oder können kontinuierlich durch Elektromotoren geöffnet und geschlossen werden. Für die Simulation dieser Arbeit wurden aufgrund der einfacheren Implementierbarkeit kontinuierlich einstellbare Ventile gewählt, die mittels eines PID-Reglers jeweils die benötigte Durchflussrate der Kabinenluft steuern. Für die Untersuchung von mehreren Varianten eines ECS-Systems ist eine allgemeingültige Beschreibung der verwendeten Ventile notwendig. Das Modell der verwendeten Ventile beschreibt die Durchflussrate in Abhängigkeit eines Durchflusskoeffizienten, dem Öffnungsgrad der Blende sowie dem öffnungsabhängigen Druckverlust.

6.2.4 Validierung und Verifikation

Die Aufgabe der Domäne *Validierung und Verifikation* besteht in der Überprüfung der Richtlinienkonformität der domänenspezifischen Tätigkeiten in dem globalen Kontext des Gesamtsystems Flugzeug. Dazu werden die Ergebnisse der Anwendungsfälle mit den ursprünglichen Anforderungen verglichen und der Erfüllungsgrad festgestellt (vgl. Kapitel 3.1.2). Bei einer Über- oder Unterschreitung der Zielvorgaben wird mit den Domänenverantwortlichen Lösungen für eine Anpassung der Systemelemente gesucht. Gerade im Bereich der *Validierung und Verifikation* ist bei häufig wiederkehrenden Tätigkeiten eine Automatisierung möglich. Dies ist unter anderem durch sogenannte Zustandsübergangsdiagramme (engl.: statecharts) in Simulink möglich, welche die Integration von komplexen Entscheidungsvorgängen in Arbeitsabläufe zulassen.

6.3 Definition der Anwendungsfälle

Nachfolgend werden auszugsweise zwei Anwendungsfälle des ECS (vgl. Tabelle 6-2) mit dem Ziel erläutert, das Vorgehen bei der Identifikation von Datenflüssen anschaulich zu beschreiben. Erst nach der Definition der zu kommunizierenden Datensätze über alle Anwendungsfälle hinweg und eine entsprechende Dokumentation in SysML-

Blockdiagrammen kann die Datenbankstruktur zum domänenübergreifenden Datenaustausch erzeugt und genutzt werden.

6.3.1 Anwendungsfall I – Druckregelung im Standardflugbetrieb

Wie bereits in Tabelle 6-2 angedeutet worden ist, bestehen aufgrund der vielfältigen Anforderungen mehrere Einsatzszenarien für ein ECS. Darunter fällt die Regelung des Drucks innerhalb der Klimazonen während einer Flugmission. Die Regelgröße Innendruck hängt dabei von der Flughöhe, dem vorgegebenen Minimaldruck sowie den maximalen Druckgradienten in Steig- und Sinkflug ab. Für die Berechnung des Systemverhaltens innerhalb des Anwendungsfalles, müssen die Ein- und Ausgangsgrößen der beteiligten Entwicklungsdomänen festgelegt und die domänenspezifischen Partialmodelle angepasst werden. Entsprechend des Informationsflusses und der zu kommunizierenden Daten sind die Schnittstellen der Partialmodelle zur Metadatenbank zu gestalten.

Tabelle 6-3 Informationsfluss Anwendungsfall I

↑ Input ↓ Output	Externe Daten	System Design	System Installation	System Simulation	Verifikation & Validierung
Wärmelasten elektrische Ausrüstung	↓	↑		↑	
Anzahl Passagiere	↓	↑		↑	
Geometrieparameter Flugzeugstruktur	↓	↑	↑	↑	
Schubmodell Triebwerk	↓	↑		↑	
Verdichterkennfeld Triebwerk	↓	↑		↑	
Systemanordnung		↓	↑	↑	
Leitungsparameter		↓	↑	↑	
Ventilparameter		↓	↑	↑	
Geometrieparameter Klimazonen			↓	↑	
Geometrieparameter Leitungen			↓	↑	
…					
Zonaler Innendruck und Druckgradient				↓	↑

Tabelle 6-3 zeigt zusammenfassend den Informationsfluss zwischen Entwicklungsdomänen für den aktuellen Anwendungsfall, für den die Schnittstellen zwischen dem Tabellenkalkulationsprogramm MS Excel, der CAD-Software CATIA V5, der Simulationssoftware Matlab/ Simulink und der MySQL-Datenbank nach dem ETL-Prozess (vgl. Kapitel 5.5) zu programmieren sind. Ein Beispiel für den Datenaustausch mittels integrierter Schnittstellen (Matlab) ist in Annex IV zu finden, ein weiteres Beispiel für den Datenaustausch über Programmierschnittstellen (CATIA V5) findet sich in Annex V. Dabei spielt die Festlegung der Datentypen und Genauigkeit eine entscheidende Rolle, um Inkompatibilitäten und Konvertierungsfehler zu vermeiden.

Ist die Grundlage der Kommunikationsfähigkeit zwischen den Entwicklungsabteilungen und dem Metamodell hergestellt, können die domänenspezifischen Modelle parallel bearbeitet werden, um die domänenübergreifenden Informationen entsprechend des definierten Anwendungsfalles zu erzeugen. Schrittweise wird nach definiertem Arbeitsablauf und festgelegtem Informationsfluss das ECS entwickelt, bis das System den Anforderungen des Anwendungsfalles genügt. Dabei wird kontinuierlich der Detailgrad sowohl des CAD- als auch des Simulationsmodells erhöht und Systemparameter wie geometrische Dimension und Leistungsbedarf berechnet. Je nach Arbeitsablauf wird zu bestimmten Meilensteinen die Verifikation und Validierung vorgenommen, um bestimmte Versionen einer Entwicklung auf ihre Anforderungserfüllung zu überprüfen. Wird an einem Meilenstein die Nichterfüllung der Anforderungen eines Anwendungsfalles festgestellt, können in interdisziplinären Teams die Lösungsmöglichkeiten diskutiert werden. Aufgrund der modellbasierten Vorgehensweise sind allen Beteiligten die Stellgrößen eines Systems bekannt und können somit unter multidisziplinären Aspekten diskutiert werden. Die Modelle stehen nachfolgend für die Untersuchung weiterer Anwendungsfälle auf Systemebene (z.B.: ATA-Kapitel) und Gesamtsystemebene (Flugzeug) zur Verfügung.

6.3.2 Anwendungsfall II – Temperaturregelung Standfall

Ein weiter Anwendungsfall innerhalb des Systementwurfs eines ECS ist die Temperaturregelung der Klimazonen im Standfall. Dazu zählen sowohl das Aufheizen als auch das Abkühlen des Kontrollvolumens innerhalb eines vorgegebenen Zeitrahmens (vgl. SAE-ARP85E, 1991). Dieser Anwendungsfall bestimmt neben dem Betrieb im Flug die installierte Leistung und damit die geometrischen Dimensionen des ECS.

Analog zum ersten Anwendungsfall werden schrittweise die domänenübergreifenden Systemparameter definiert und damit der kontinuierliche Aufbau des Metamodells und der Datenbankstruktur vorangetrieben. Aus den Anforderungen an den Anwendungsfall geht hervor, dass andere Parameter für eine Systemuntersuchung benötigt werden, als in Kapitel 6.3 beschrieben. Es werden beispielsweise keine Wärmelasten von elektrischer Ausrüstung und Passagieren berücksichtigt. Weiterhin ist für die Simulation zusätzlich zu dem Modell des Flugtriebwerks je ein Modell für die Zapfluftversorgung durch eine APU und gegebenenfalls eine bodengebundene, stationäre Quelle notwendig. Tabelle 6-4 zeigt ausschnittsweise Systemparameter des ECS und die Verwendung der Parameter in den eben skizzierten Anwendungsfällen.

Tabelle 6-4 Informationsfluss Anwendungsfälle I & II

↑ Input ↓ Output	Externe Daten	System Design	System Installation	System Simulation	Verifikation & Validierung	Use Case I	Use Case II
Wärmelasten elektrische Ausrüstung	↓	↑		↑		□	□
Anzahl Passagiere	↓	↑		↑		□	□
Geometrieparameter Flugzeugstruktur	↓	↑	↑	↑		□	□
Schubmodell Triebwerk	↓	↑		↑		□	□
Verdichterkennfeld Triebwerk	↓	↑		↑		□	□
Verdichterkennfeld APU	↓	↑		↑			□
Systemanordnung		↓	↑	↑		□	□
Leitungsparameter		↓	↑	↑		□	□
Ventilparameter			↑	↑		□	□
Geometrieparameter Klimazonen			↓	↑		□	□
Geometrieparameter Leitungen			↓	↑		□	□
…							
Zonaler Innendruck und Druckgradient				↓	↑	□	□

Nachdem die Elemente eines Systems mit allen relevanten Eigenschaften definiert sind, kann in Anlehnung an Kapitel 5.3 die Struktur der Metadatenbank erzeugt werden. Abbildung 6-8 stellt ausschnittsweise die automatisch generierte Datenbankstruktur dar, die nun als Basis für die domänenübergreifende Kommunikation verwendet werden kann.

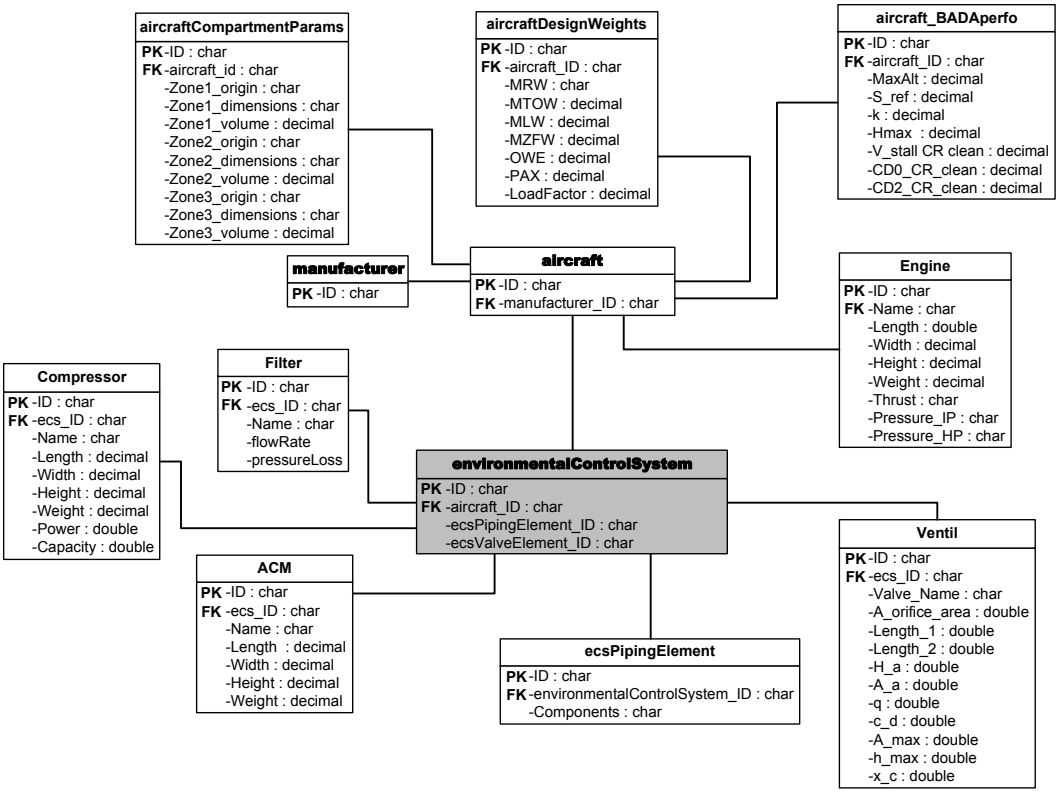


Abbildung 6-8 Ausschnitt der SQL-Datenbankstruktur des Metamodells

6.4 Parallelisierung und Automatisierung

Die Parallelisierung von Abläufen erfordert die Entkopplung der zugrundeliegenden Entwicklungstätigkeiten. In sequentiellen Abläufen erzeugt eine Tätigkeit einen Datensatz, von dem eine andere Tätigkeit abhängig ist und somit zeitlich versetzt stattfinden muss. Wird die Abhängigkeit der beiden Abläufe aufgelöst, besteht die Möglichkeit der parallelen Bearbeitung. Diese Entkopplung von Entwicklungstätigkeiten wird über die Metaebene ermöglicht, da die beteiligten Arbeitsabläufe nur noch indirekt über das Systemmodell kommunizieren. Nach dem Erzeugen einer „Startlösung“ (Startdatensatz durch Instanziierung des Metamodells) können alle Domänen parallel an der Detaillierung des Modells arbeiten.

Innerhalb dieser Arbeit wird ein eng gekoppeltes Metamodell verwendet, bei dem jede Domäne über den gesamten Metadatensatz Leserechte besitzt, jedoch nur für bestimmte Zellen entsprechende Schreibrechte. Dadurch werden inkonsistente Datensätze vermieden. Nachdem in dem Metamodell eines Systems lediglich Daten abgelegt werden, die bereichsübergreifenden Nutzen besitzen, kann es während der Entwicklung dennoch zur inkonsistenten Benutzung von Daten kommen. Ändert beispielsweise Domäne A einen Datensatz, mit dem eine andere Domäne B gerade arbeitet, müssen entsprechende Mechanismen vorhanden sein, die Inkonsistenz zu beheben. Dazu bestehen zwei Möglichkeiten: zum einen könnte das Partialmodell der Domäne B in bestimmten Abständen die verwendeten Daten mit den Daten des Metamodells abgleichen (*periodisch* oder *anfragegesteuert*); zum anderen könnte Domäne B über eine Änderung des Datensatzes informiert werden (*ereignisgesteuert*). Die beschriebene Änderung von Datensätzen wird in der industriellen Anwendung von PDM-Systemen im Rahmen des Änderungswesens ermöglicht, wodurch eine formale Vorgehensweise zur Veränderung der Eigenschaften eines Attributs definiert ist. Aufgrund des Implementierungsaufwandes wurde innerhalb dieser Arbeit bewusst auf die Versionierung und Freigabe von Instanzen des Metamodells verzichtet. Dennoch wird auf die Wichtigkeit der Datenverwaltung der Systemmodelle aus Gründen der Vollständigkeit verwiesen.

Vielmehr ist es das Ziel des Anwendungsbeispiels, die Automatisierung von interdisziplinären Arbeitsabläufen auf Basis des Metamodells zu realisieren. Dabei wurden zunächst innerhalb des ersten Anwendungsfalles *Druckregulierung Standardflugbetrieb* auf Basis des ETL-Prozesses die Skripte erstellt, welche die Schnittstellen zwischen den Partialmodellen und dem Metamodell darstellen. Durch Ausführung eines dieser Skripte werden die Daten eines Partialmodells automatisch in das Metamodell geschrieben oder auf umgekehrtem Wege ausgelesen und in das Partialmodell geladen.

Der automatisierte Datenaustausch über das Metamodell hat den Vorteil, dass eine manuelle Interpretation der Partialmodelle (Extraktion, Transformation und Laden der benötigten Daten) entfällt. Weiterhin stehen die Datensätze stets im richten Format zur Verfügung und können ohne Umwege weiterverwendet werden. Die Schnittstellen können dabei sowohl von den Nutzern der Partialmodelle „per Hand“ ausgeführt oder in automatisierten Abläufen aufgerufen werden.

Innerhalb des Anwendungsfalles *Druckregulierung Standardflugbetrieb* wird auf Basis der automatischen Schnittstellen die komplette Automatisierung von domänenspezifischen und domänenübergreifenden Arbeitsabläufen demonstriert:

Automatisierung domänenspezifischer Arbeitsabläufe

Innerhalb der Domäne *Verifikation & Validierung* wird die Automatisierung der Fehlerberichterstellung realisiert: Nach dem Auslesen der benötigten Datensätze zur Überprüfung des Anforderungserfüllungsgrades werden automatisch entsprechende Berichte generiert, die danach zur weiteren Auswertung zur Verfügung stehen. Zur Veranschaulichung wird ein charakteristischer Arbeitsablauf in der ECS-Entwicklung skizziert: Nach der Erzeugung einer Startlösung des Metamodells erfolgt die Erstellung des ersten DMU und damit erste Machbarkeitsanalysen der physikalischen Systemintegration (*System Installation*). Ebenfalls parallel werden erste Simulationsmodelle entsprechend der Systemarchitektur erstellt, um das dynamische Systemverhalten antizipieren zu können (*System Simulation*). Ein Parameter, der in allen Partialmodellen Anwendung findet, ist beispielsweise der Durchmesser der Klimahauptleitungen. Aufgrund der erzeugten Startlösungen für den Durchmesser werden sowohl bei der Systemintegration in das DMU als auch bei der Systemsimulation Fehler oder Regelverletzungen festgestellt. Diese Fehlermeldungen erzeugen entsprechende Einträge im Metamodell, die wiederum zur automatisierten Fehlerberichterstellung in der Domäne *Verifikation & Validierung* genutzt werden. Der Ablauf des Vorgehens von der Erzeugung einer Startlösung in der Systemdatenbank bis hin zur automatischen Generierung eines XML-basierten Fehlerberichts ist in Abbildung 6-9 dargestellt. Der Fehlerbericht kann mit jedem gängigen HTML-Browser geöffnet werden und innerhalb der Domänen kommuniziert werden.

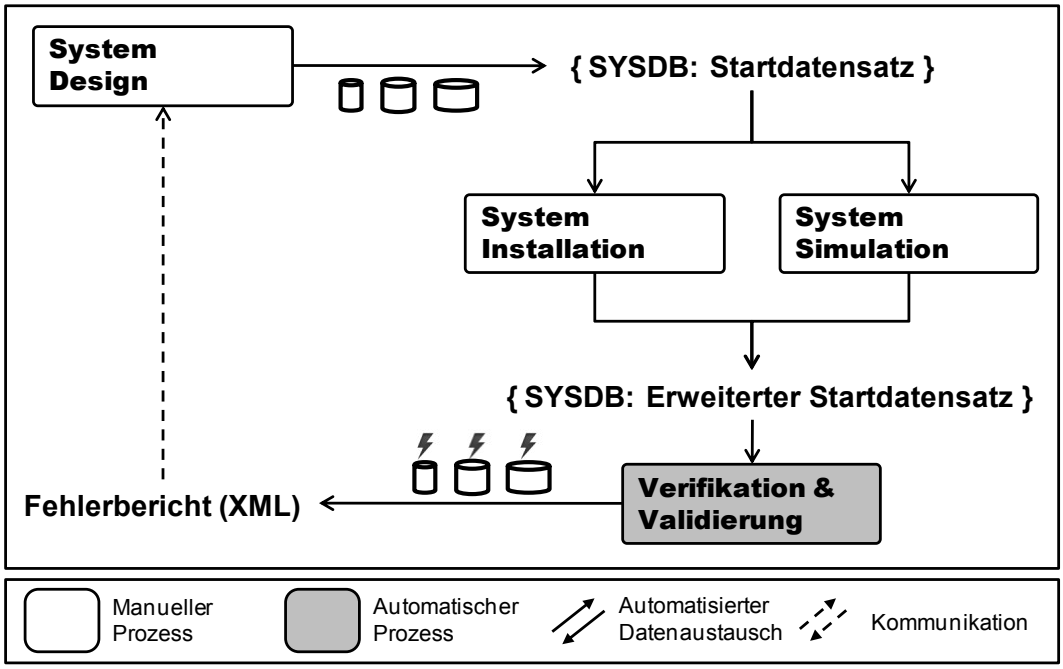


Abbildung 6-9 Automatisierung domänenspezifischer Arbeitsabläufe

Automatisierung domänenübergreifender Arbeitsabläufe

Ein Beispiel für die Automatisierbarkeit von domänenübergreifenden Tätigkeiten ist innerhalb des zweiten Anwendungsfalles *Temperaturregelung Standfall* implementiert. Für diese Anwendung wird innerhalb der Domäne *System Simulation* eine Optimierung des Leitungsquerschnitts der Klimahauptleitung durchgeführt und der resultierende Rohrdurchmesser an das Metamodell übergeben. Um die Plausibilität des Ergebnisses im Gesamtkontext Flugzeug zu überprüfen, ist die Einbauuntersuchung und das Kollisionsverhalten mit weiteren Systemen innerhalb des DMU notwendig. Dazu wird innerhalb der Domäne *System Simulation* das DMU in einem Stapelverarbeitungsmodus (engl.: batch mode) gestartet, der Rohrdurchmesser als Parameter an das Modell übergeben und die Kollisionsanalyse ausgeführt. Nach Beendigung der Stapelverarbeitung steht im Metamodell die Information zur Verfügung, ob Kollisionen mit anderen Systemen vorhanden sind und sich damit das erzeugte Ergebnis für eine Weiterverwendung eignet. Für die Bereitstellung dieser Information sind keine personellen Ressourcen der Domäne *System Installation* notwendig, wodurch effizientere Abläufe in allen beteiligten Fachabteilungen ermöglicht werden.

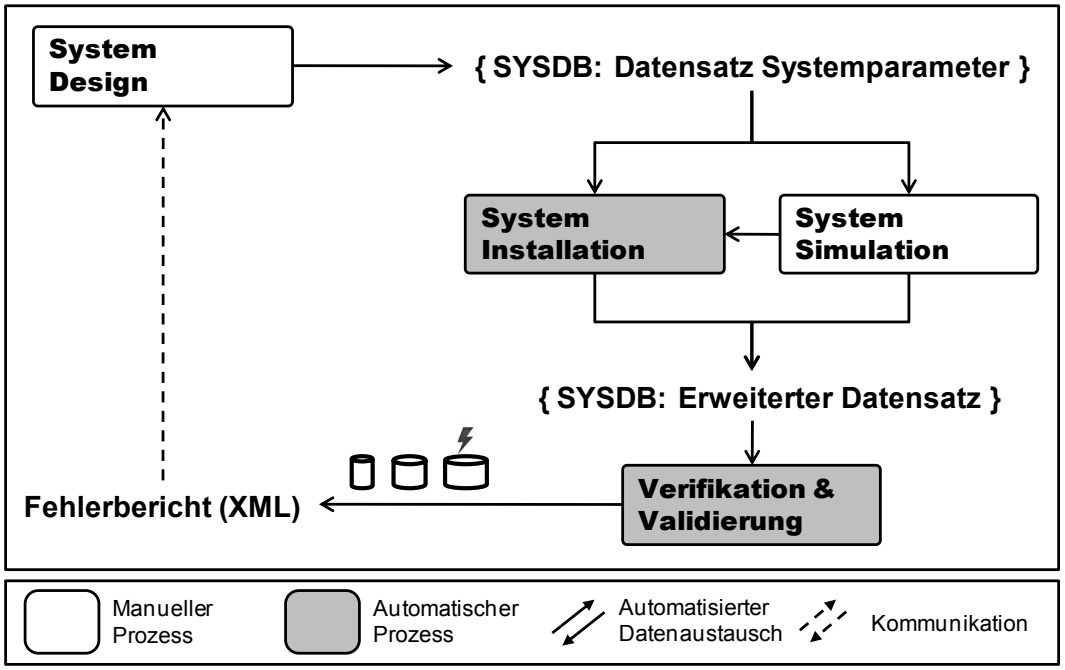


Abbildung 6-10 Automatisierung domänenübergreifender Arbeitsabläufe

6.5 Ergebnis des modellbasierten Anwendungsbeispiels

Anhand der Regularien in Bezug auf Druck- und Temperaturregelung werden verschiedene Anwendungsfälle definiert, auf deren Basis der Entwurf des Systems und die Festlegung der dimensionierenden Größen des Systems erfolgt. Jeder Anwendungsfall ist charakterisiert durch einen repräsentativen Arbeitsablauf und domänenspezifische, an den Anwendungsfall adaptierte Partialmodelle. Sukzessive können auf Basis der Metadatenbank die an der Entwicklung beteiligten Domänen Anwendungsfälle bearbeiten, wobei eine kontinuierliche Verifikation und Validierung anhand eines Meilensteinplanes erfolgt.

Durch die Verwendung des domänenübergreifenden Metamodells (in Form einer relationalen Datenbank) wird die Kommunikation zwischen verschiedenen Fachgebieten und den hierin verwendeten Softwarewerkzeugen ermöglicht. Die zugrunde liegende Methodik berücksichtigt jedoch nicht ausschließlich die Gestaltung von Softwareschnittstellen, sondern explizit die integrative Betrachtung von Arbeitsabläufen der Systementwicklung. Je nach Entwicklungsaufgabe oder Entwicklungsumgebung stellt das Metamodell die Basis für semi- oder vollautomatisierte, multi-disziplinäre Tätigkeiten dar. Die Möglichkeiten zur Parallelisierung und Automatisierung von Arbeitsabläufen wird innerhalb der Anwendungsfälle I und II an Beispielen demonstriert.

Abbildung 6-11 zeigt das Ergebnis der Systemsimulation innerhalb des Anwendungsfalles *Druckregulierung Standardflugbetrieb* als Interpretationsgrundlage im Hinblick auf die Richtlinienerfüllung der ECS-Funktion. In der Abbildung ist zu erkennen, dass

der Druckgradient innerhalb der behördlichen Vorgaben verbleibt, und der Innendruck durch die Regelung der Klimaanlage nie unter die Referenzhöhe von 8000ft (entspricht 75262Pa bei ISA+0K) sinkt.

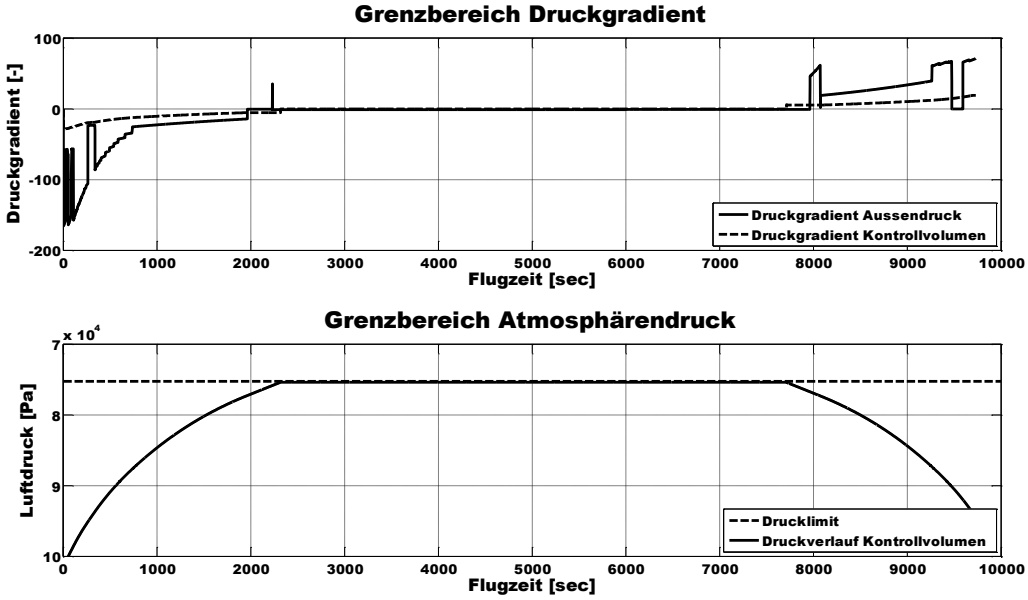


Abbildung 6-11 Grenzbereich für Druck & Druckgradient in den Klimazonen

Für den Anwendungsfall *Temperaturregelung Standfall* werden die Ergebnisse der Systemsimulation in Abbildung 6-12 dargestellt. Die Kurven stellen den Temperaturverlauf innerhalb des Kontrollvolumens der Kabine unter den geforderten Randbedingungen dar. Es ist zu erkennen, dass sowohl im APU-Betrieb als auch unter Verwendung der Triebwerke die geforderten Temperaturänderungen in weniger als 30 Minuten zu erreichen sind (vgl. ARP85E, 1991).

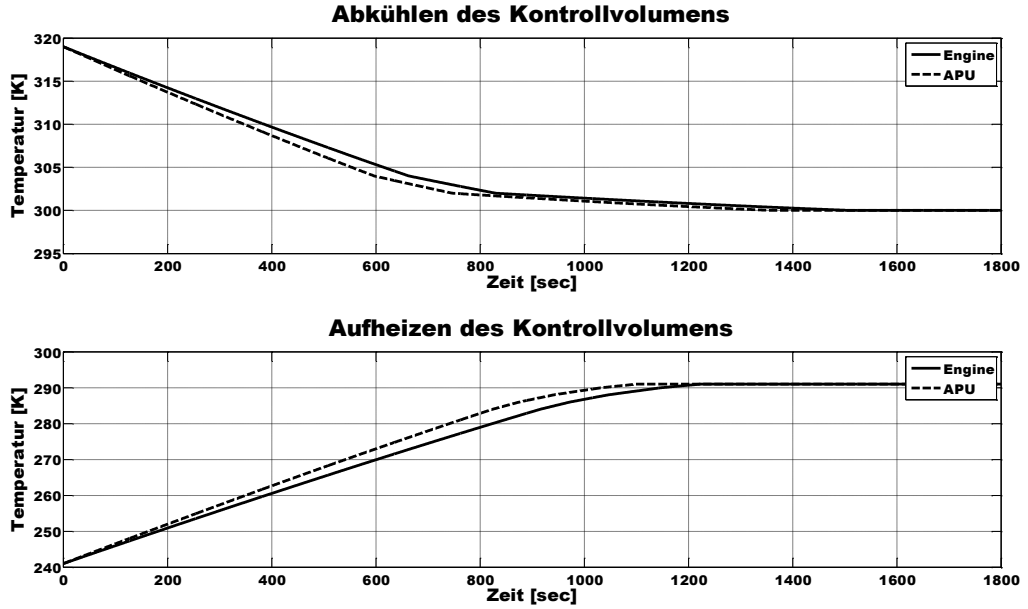


Abbildung 6-12 Simulation des Temperaturverhaltens des Kontrollvolumens

Die Ergebnisse des Entwurfs eines voll parametrischen ECS-Modells am Beispiel einer A320 ähnlichen Flugzeugarchitektur stimmen weitestgehend mit den realen Leistungsdaten der A320 ECS überein. Eine weitere Detaillierung des Simulationsmodells zur Erzielung exakterer Ergebnisse erscheint nur unter Zuhilfenahme einer breiteren Datenbasis innerhalb industrieller Untersuchungen sinnvoll. Vielmehr war es das Ziel des Anwendungsbeispiels, die Implementierungsfähigkeit und Vorteile des modellbasierten Vorgehens zu demonstrieren sowie die industrielle Anwendbarkeit mit einem praxisnahen Anwendungsbeispiel zu unterstreichen.

Das vorgestellte Vorgehen entspricht den Prämissen (vgl. Kapitel 5), die zur erfolgreichen Implementierung eines modellbasierten Ansatzes zur Entwicklung Flugzeugsystemen berücksichtigt werden sollten. Durch die durchgängige Abbildung von Anforderungen und resultieren Produktfunktionen wird die Absicherung der Anforderungserfüllung gewährleistet. Die Definition von Systemarchitekturen mit SysML erlaubt die fachbereichsübergreifende Abbildung der Gesamtsystemstruktur und dient damit zur Definition von Schnittstellen zwischen Domänen unter Einbezug aller Domänenbeteiligten. Über die Festlegung der domänenübergreifen Datenflüsse entsteht sukzessive das Metamodell und kann in eine relationale Datenbank überführt werden, die nach der Bereitstellung von Schnittstellendefinitionen zwischen Partialmodellen und Metadatenbank zum Datenaustausch verwendet wird. Mit der geschaffenen Infrastruktur steht den Systementwicklern das Grundgerüst für Arbeitsabläufe zur Verfügung, die zu einem größeren Maß als bisher parallelisiert und automatisiert werden können.

7 ZUSAMMENFASSUNG UND AUSBLICK

7.1 Ergebnisse der Arbeit

Die steigenden Anforderungen an luftfahrttechnische Produkte in Bezug auf Kosten und Qualität erhöhen den Druck auf aktuelle Entwicklungsprozesse. Die geforderte Effizienzsteigerung von Entwicklungsabläufen kann unter anderem durch die gesteigerte Parallelisierung und Automatisierung von domänenspezifischen Tätigkeiten erreicht werden. Vor diesem Hintergrund war das Ziel der vorliegenden Arbeit die Schaffung einer Methode für die effiziente Entwicklung von Flugzeugsystemen durch die tiefere Integration der beteiligten Entwicklungsdomänen sowie eine stärkere Parallelisierung und Automatisierung von Entwicklungsabläufen. Die Gestaltung des Vorgehensmodells der Methode erfolgt dabei unter Berücksichtigung von industriellen Randbedingungen.

Um die Grundlagen für einen methodischen Ansatz zu schaffen, werden zunächst die Grundlagen der Flugzeugentwicklung sowie des Entwurfs von Flugzeugsystemen diskutiert und aktuelle Defizite auf globaler, prozeduraler und technischer Ebene innerhalb von luftfahrttechnischen Entwicklungsumgebungen identifiziert. Weiterhin wird ein generischer Entwicklungsablauf vorgestellt, anhand dessen zum einen die Problematik aktueller Entwicklungsprozesse veranschaulicht und zum anderen die Anwendbarkeit der vorgestellten Methode demonstriert werden.

Für die Erstellung einer fundierten und substantiellen Methode werden aktuelle Methoden der Systementwicklung analysiert und dabei zwischen deskriptiven und präskriptiven Methoden der Produktentwicklung und Prozessgestaltung unterschieden. Die Erfolgsfaktoren zukünftiger Entwicklungsprozesse sind das Ergebnis dieser Analyse, wobei zwischen Faktoren der Prozessunterstützung und Modellierungsunterstützung unterschieden wird. Von maßgeblicher Bedeutung ist die Unterstützung der Konzeptphase einer Entwicklung, um Gestaltungsentscheidungen zu einem möglichst frühen Zeitpunkt treffen zu können und dabei eine größere Entscheidungsrobustheit zu erreichen. Besonders die Automatisierung von Informations- und Datenflüssen zwi-

schen allen am Entwicklungsprozess Beteiligten und die Digitalisierung von Daten und Wissen bieten großes Potenzial, um dieses Ziel zu erreichen. Weiterhin hilft die Unterstützung einer durchgängigen Modellierung von Anforderungen zu verbessertem Systemverständnis der Entwickler und zu robusteren Architekturen.

Auf Basis der Grundlagen werden vorhandene Modellierungsansätze auf deren Eignung für die Problemstellung dieser Arbeit und die Erstellung eines Systemmodells hin diskutiert. Die Modellierung von Systemen auf Metaebene wird als Grundlage zur Realisierung der genannten Erfolgsfaktoren identifiziert. Basierend auf dieser Erkenntnis wird ein formales Metamodell zur Beschreibung von Flugzeugsystemen hergeleitet. Das Metamodell basiert auf dem Prinzip der engen Kopplung von relevanten Ausschnitten der domänenspezifischen Partialmodelle und dient zur Beschreibung von Systemelementen und deren Relationen. Die Attributierung der Elemente erfolgt nach den Prinzipien der Typisierung und Instanziierung, was eine Umsetzung in gängigen Modellierungswerkzeugen ermöglicht.

Unter Diskussion von Alternativen zur Implementierung des Metamodells in vorhandene Unternehmensarchitekturen, wird eine konkrete Möglichkeit zur rechnertechnischen Umsetzung erarbeitet. Dabei steht zunächst die integrative Modellierung von Anforderungen, Funktionen sowie Systemelementen und deren Relationen im Vordergrund. Ziel der Systemmodellierung unter Berücksichtigung von Anwendungsfällen ist die Absicherung der Systemfunktion und der Robustheit der untersuchten Systemarchitekturen. SysML wurde als Modellierungssprache gewählt, da sie bereits einen auf Systementwickler zugeschnittenen Funktionsumfang der UML darstellt. Die graphische Orientierung und die vielfältigen Schnittstellen gängiger SysML-Werkzeuge garantieren die Anwendbarkeit und Umsetzbarkeit der Methode. Ist ein System auf der Metaebene modelliert, kann es in eine relationale Datenbankstruktur übersetzt werden, welche die Basis für den domänenübergreifenden Datenaustausch darstellt. Relationale Datenbanken sind durch den gut strukturierten Syntax der SQL leicht zu bedienen und besitzen aufgrund ihrer weiten Verbreitung den Vorteil, dass kommerzielle Software häufig Schnittstellen zum SQL-Datentransfer bereitstellt.

Abschließend wird die Anwendbarkeit des modellbasierten Vorgehens exemplarisch an der Nachprojektierung einer Flugzeugklimaanlage demonstriert. Am Beispiel eines Referenzflugzeuges (ähnlich dem Airbus A320) wird eine Flugzeugklimaanlage als repräsentatives mechatronisches Flugzeugsystem unter Berücksichtigung behördlicher Anforderungen ausgelegt. Die aus den Anforderungen resultierenden Anwendungsfälle werden für die Demonstration des Potenzials der modellbasierten Vorgehensweise zur Automatisierung und Parallelisierung von Arbeitsabläufen verwendet. Zum einen

wird erfolgreich demonstriert, dass der Datenaustausch zwischen zwei Domänen über das Metamodell genauso automatisiert werden kann wie dies mit klassischen Schnittstellen oder neutralen Datenformaten der Fall ist. Innerhalb des modellbasierten Vorgehens reduziert sich allerdings der ausgetauschte Datensatz auf tatsächlich benötigte Datenpakete und reduziert damit das Datenaustauschvolumen. Zum anderen zeigte das Beispiel auf, dass vormals sequentielle Arbeitsabläufe durch Auflösen der Datenschnittstelle entkoppelt werden können und die Nutzung eines Metamodells zur stärkeren Parallelisierung von Tätigkeiten dienen kann. Ebenso wurde die komplette Automatisierbarkeit von domänenübergreifenden Tätigkeiten demonstriert, was in der industriellen Anwendung zu erheblichen Zeiteinsparungen bei häufig wiederkehrenden Aufgaben führen kann. Dabei erfolgte der Einsatz der vorgestellten Methode unter den identifizierten Prämissen und den Anforderungen an Modelle.

Dementsprechend beschreibt die in dieser Arbeit entwickelte Methode ein Konzept für ein durchgängiges Systemmodell, das insbesondere spezifische Sichtweisen auf ein System unterstützt und unter Berücksichtigung von zugrundeliegenden Prozessen sowie der Unternehmensstruktur, den aktuellen Stand der Entwicklung luftfahrttechnischer Produkte zu verbessern in der Lage ist. Der Wegfall von manuellem Informationsaustausch zwischen Modellen und des damit verbundenen Koordinationsaufwands erleichtert den iterativen Systementwurf dabei erheblich. Das Parallelisieren von Entwicklungstätigkeiten wird maßgeblich ermöglicht durch die Identifikation von Informationsflüssen und der Beseitigung von Inkompatibilitäten zwischen Datenformaten. Dadurch werden die Minimierung des Kommunikationsaufwandes und die Automatisierung von Arbeitsabläufen ermöglicht. Die Erarbeitung des formalen Systemmodells als wesentlicher Bestandteil einer modellbasierten Methode sowie die Implementierung der Methode innerhalb einer konkreten Software- und Hardwareinfrastruktur stellen den Schwerpunkt dieser Arbeit dar.

7.2 Ausblick

Eine weitere Untersuchung der vorgestellten Methode und der bisher erzielten Ergebnisse ist unter verschiedenen Aspekten möglich, wobei die Weiterentwicklung der Methode sowie die Anwendung der Methode zur Diskussion stehen.

Zunächst sind die Modellierungskonstrukte des Metamodells vor dem Hintergrund einer industriellen Anwendung auf ihre Vollständigkeit hin zu untersuchen und gegebenenfalls zu verfeinern oder zu ergänzen. Weiterhin ist zu überprüfen, ob die in einer relationalen Datenbank darstellbaren Datentypen zur Modellierung aller Aspekte von Flugzeugsystemen ausreichend sind und unter welchen Gegebenheiten die Verwen-

dung von objektorientierten Datenbanken der zusätzliche Modellbildungsaufwand gerechtfertigt wird. Ebenso relevant ist eine detaillierte Untersuchung der Integration der Systemmodelle in Produktdatenmanagementsysteme, um die erfolgreiche Verwaltung und Zugriffssteuerung der domänenspezifischen Partialmodelle auf das Metamodell sicherzustellen.

Im Bereich der Flugzeugsystementwicklung sollte die Methodenanwendung auf das Gesamtsystem Flugzeug erweitert werden und zukünftige Flugzeugsystemarchitekturen unter multidisziplinären Aspekten entworfen werden. Ein Beispiel für eine solche Anwendung wäre die Untersuchung alternativer, verteilter Energiequellen eines Flugzeuges. Im Rahmen einer derartigen Untersuchung sollten die Skalierbarkeit des Ansatzes auf reale Anwendungen innerhalb der Industrie sowie potenzielle Grenzen der Integration einer Vielzahl von (verteilten) Domänen analysiert werden.

Nicht zuletzt kann die erarbeitete Methode in weitere Fachgebiete übertragen werden. Alternative Bereiche des Maschinenwesens, wie der Automobilbau kommen hierfür gleichermaßen in Frage, wie das Anwendungsfeld der anwendungsorientierten Softwareentwicklung.

LITERATURVERZEICHNIS

ACARE, 2004. *Strategic Research Agenda - Volume 1*. Advisory Council for Aeronautics Research Europe. Brüssel.

ANDERL, R., 2011. *Virtuelle Produktentwicklung B: Produktdatenmanagement*. Darmstadt.

ARNOLD, V.; DETTMERING, H.; ENGEL, T. & KARCHER, A., 2011. *Product Lifecycle Management beherrschen: Ein Anwenderhandbuch für den Mittelstand.* 2. Aufl. Berlin: Springer Berlin.

ARP1270B, 2010. *Aircraft Cabin Pressurization Control Criteria*. SAE - Society of Automotive Engineers.

ARP4754A, 2010. *Guidelines for Development of Civil Aircraft and Systems*. SAE - Society of Automotive Engineers.

ARP85E, 1991. *Aircraft Pressurization Control Criteria*. SAE - Society of Automotive Engineers.

ARTWORK, 2003. *Das Produkt- und Prozessmodell der Engineering Workbench: Abschlußbericht*. Online im Internet: URL: http://artwork.in.tum.de/ [Stand 2011-12-01].

BANDOW, G. & HOLZMÜLLER, H. 2010. *"Das ist gar kein Modell!": Unterschiedliche Modelle und Modellierungen in Betriebswirtschaftslehre und Ingenieurwissenschaften.* 1. Aufl. Wiesbaden: Gabler.

BAUER, A. & GÜNZEL, H., 2009. *Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung.* 3. Aufl. Heidelberg: dpunkt.

BRANDSTÄTTER, M. F., 2009. *Kompatibilitätsmodellierung im Systems-Engineering-Umfeld.* 1. Aufl. München: Verl. Dr. Hut.

BROY, M.; FEILKAS, M.; HERRMANNSDOERFER, M.; MERENDA, S. & RATIU, D., 2010. Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments. *4. Proceedings of the IEEE (98)*, Seiten 526–545.

BROY, M., 2011. *Requirements Engineering. Skript zur Vorlesung*. Technische Universität München. Institut für Informatik.

BUNGARTZ, H.-J; ZIMMER, S.; BUCHHOLZ, M. & PFLÜGER, D., 2009. *Modellbildung und Simulation: Eine anwendungsorientierte Einführung*. Berlin: Springer.

BUTTERWORTH-HAYES, P., 2009 All-electric aircraft research speeds up: *Aerospace America*, Ausgabe Januar 2009, S.4–7.

BUUR, J., 1990. *A theoretical approach to mechatronic design.* PhD thesis. Technical University of Denmark. Institute for Engineering Design. Lyngby, Denmark.

CHEN, P. P.-S., 1976 The Entity-Relationship Model – Toward a Unified View of Data: *ACM Transactions on Database Systems*, Vol. 1, No. 1. Mar. 1976, S. 9–36.

CZARNECKI, K. & HELSEN, S. 2006. Feature-based survey of model transformation approaches. *IBM Systems Journal Nr. 45*. S. 621–645.

DASSAULT SYSTEMES, 2011. *CATIA V5R21: Fact Sheet*. URL: http://www.3ds.com/fileadmin/PRODUCTS/CATIA/PDF/c521_factsheet.pdf [Stand 2011-11-30].

DAVIES, M., 2003. *The standard handbook for aeronautical and astronautical engineers*. New York: McGraw-Hill.

DEROUINEAU, J.-L., 2009. Power Optimized More Electrical Aircraft, in Masaryk University, Brno C. (Hg.): *Proceedings of the European conference TOWARDS eENVIRONMENT*, S. 293–295.

DOLEZAL, W. R., 2008. *Success Factors for Digital Mock-ups (DMU) in Complex Aerospace Product Development*. Dissertation, Technische Universität München. Lehrstuhl für Luftfahrttechnik.

DOUMBIA, F.; LAURENT, O.; ROBACH, C. & DELAUNAY, M.; 2010. Relying on Testability Concepts to ease Validation and Verification activities of AIRBUS Systems. *International Journal on Advances in Software*, Vol. 3 No. 1 & 2, S. 41–53.

EGELHOFER, R., 2009. *Aircraft design driven by climate change*. 1. Aufl. München: Verl. Dr. Hut.

EHRLENSPIEL, K., 2007. *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit.* 3. Auflage, München: Hanser.

EHRLENSPIEL, K.; KIEWERT, A. & LINDEMANN, U., 2007. *Kostengünstig Entwickeln und Konstruieren: Kostenmanagement bei der integrierten Produktentwicklung.* 6., überarbeitete und korrigierte Auflage.
Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

ERICSSON, A. & ERIXON, G. 1999. *Controlling design variants: Modular product platforms*. New York: Society of Manufacturing Engineers; ASME Press.

FAA, 2011. *Part 25: Airworthiness Standards: Transport Category Airplanes*. URL: http://www.faa.gov/aircraft/air_cert/airworthiness_certification/std_awcert/std_awcert_regs/regs/ [Stand 2011-11-21].

FALEIRO, L., 2006. Power Optimised Aircraft: A keystone in European research in More Electric Aircraft Equipment Systems. *Aerodays*.
Wien, Österreich 20 Jun. 2006.

FENVES, S.; FOUFOU, S.; BOCK, C.; SRIRAM, R., 2005. *CPM: A Core Product Model for Product Data*. NIST - National Institute of Standards and Technology.

FIGLAR, B. & MOHR, B., 2011. *Knowledge-based strategies in electrical aircraft wiring*. Conference of the European Aerospace Societies (CEAS).
Venedig, Italien 24. Okt. 2011.

FRIEDENTHAL, S., MOORE, A. & STEINER, R., 2009. *A practical guide to SysML: The systems modeling language.* Amsterdam: Morgan Kaufmann Object Management Group/Elsevier.

FRIEDRICH, M. O. 2010. *Funktionsorientiertes Konzept zur Unterstützung früher Phasen der Produktentwicklung in der Informationstechnik. Dissertation*.
Technische Universität München. Informationstechnik im Maschinenwesen.

GÖPFERT, J., 2009. *Modulare Produktentwicklung: Zur gemeinsamen Gestaltung von Technik und Organisation ; Theorie, Methodik, Gestaltung.*
2. Aufl. Norderstedt: Books on Demand.

GÜNZLER, A. R., 2005. *Integrationskonzepte in der modellbasierten Produktentwicklung. Dissertation*. Technische Universität München. Institut für Informatik.

HORNUNG, M., 2011. *Luftfahrtsysteme. Skript zur Vorlesung*. Technische Universität München. Lehrstuhl für Luftfahrtsysteme.

HÜNECKE, K., 2008. *Die Technik des modernen Verkehrsflugzeuges.*
1. Aufl. Stuttgart: Motorbuch-Verlag.

IABG, 2009. *Teil 1: Grundlagen des V-Modells: V-Modell XT 1.3*. Industrieanlagen-Betriebsgesellschaft mbH. URL: http://v-modell.iabg.de [Stand 2011-12-02].

INCOSE, 2006. *Systems Engineering Handbook: A guide for system lifecycle processes and activities. Version 3*.
International Council on Systems Engineering.

IRLINGER, R., 1999. *Methoden und Werkzeuge zur nachvollziehbaren Dokumentation in der Produktentwicklung.* Dissertation Technische Universität München. Aachen: Shaker 1999 (Konstruktionstechnik München, Band 31).

ISERMANN, R., 2008. *Mechatronische Systeme: Grundlagen*. Berlin, Heidelberg: Springer.

ISO, 1994. *ISO 10303-1: Industrial automation systems and integration - Product data representation and exchange: Part 1: Overview and fundamental principles.* Berlin: Beuth.

ISO, 2005. *ISO 10303-25: Industrial automation systems and integration - Product data representation and exchange: Part 25: Implementation methods: EXPRESS to XMI binding.* Berlin: Beuth.

ISO, 2007. *ISO 10303-28: Industrial automation systems and integration - Product data representation and exchange: Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas.* Berlin: Beuth.

JÄNKER, P., 2007. Mechatronics in Aerospace. *EADS Engineering Europe Symposium.* Budapest, Ungarn 8. Mai 2007

JECKLE, M., 2004. *Konzepte der Metamodellierung – Zum Begriff Metamodell*. DaimlerChrysler Forschungszentrum Ulm.

KATZKE, U. 2008. *Spezifikation und Anwendung einer Modellierungssprache für die Automatisierungstechnik auf Basis der Unified Modeling Language (UML)*. Kassel: Kassel University Press GmbH.

KERSTEN, W., 2002. *Vielfaltsmanagement: Integrative Lösungsansätze zur Optimierung und Beherrschung der Produkte und Teilevielfalt*. München: TCW Transfer-Centrum.

KIRBY, M. R., 2001. *A methodology for technology identification evaluation, and selection in conceptual and preliminary aircraft design.* Dissertation. Georgia Institute of Technology - School of Aerospace Engineering. Atlanta, USA.

KLEINOD, E. 2006. *Modellbasierte Systementwicklung in der Automobilindustrie: Das MOSES- Projekt*. Dortmund: Fraunhofer Institut für Software- und Systemtechnik.

KOSSIAKOFF, A.; SWEET, S.; SEYMOUR, J.; BIEMER, S., 2011. *Systems engineering principles and practice.* 2. Aufl. Hoboken, N.J: Wiley.

KRAUS, P. K., 2005. *Plattformstrategien - Realisierung einer varianz- und kostenoptimierten Wertschöpfung*. Bamberg: Meisenbach.

KRAUSE, F.-L., 2007. *Innovationspotenziale in der Produktentwicklung*. München: Hanser.

KÜHLBERGER, C., 2002. *Von der Prozess-Analyse zur Workflow-Unterstützung des Produktentwicklungsprozesses im Maschinenbau.* Diplomarbeit. Technische Universität München. Fakultät für Informatik.

KWIATKOWSKI, M., 2006. *Simulation of Components from the Environmental Control System. Project Report*. HAW Hamburg - Department of Automotive and Aeronautical Engineering, Hamburg.

LANGERMANN, R., 2009. *Beitrag zur durchgängigen Simulationsunterstützung im Entwicklungsprozess von Flugzeugsystemen.* Disseration. Technische Universität Braunschweig. Institut für Luft- und Raumfahrtsysteme.

LEHLE, W., 2006. *Konzeption und Entwicklung von Flugzeugklimatisierungsanlagen.* Praxis-Seminar Luftfahrt: Air Systems. Hamburg.

LERCHER, B., 2008. *Konzeption und System einer Integrationsplattform zur Entwicklung von Werkzeugmaschinen.* Dissertation. Technische Universität München. Lehrstuhl für Werkzeugmaschinen und Fertigungstechnik.

LINDEMANN, U., 2011. *Methoden der Produktentwicklung.* Umdruck zur Vorlesung WS 2011/12. Technische Universität München. Lehrstuhl für Produktentwicklung.

LINDEMANN, U.; MAURER, M. & BRAUN, T., 2009. *Structural Complexity Management: An Approach for the Field of Product Design*. Berlin, Heidelberg: Springer.

LISCOUET-HANKE, S., 2008. *A Model-Based Methodology for Integrate Preliminary Sizing and Analysis of Aircraft Power System Architectures.* Doctoral Thesis. Université Paul Sabatier. Laboratoire de Génie Mechanique de Toulouse.

LIZARRALDE, I., 2007. *Aide au pilotage d'activites d'ingenierie pour le development distribue d'un systeme complexe*. Institut National des Sciences Appliquées de Toulouse.

MAIER, J. & MATTHES, F., 2011. *Einführung in die Softwaretechnik.* Skript zur Vorlesung. Technische Universität München. Fakultät für Informatik. Lehrstuhl für Software Engineering betrieblicher Informationssysteme.

MAUL, F., 2009. *Analysis of Airbus avionics development processes and conception of tool supported processes.* Diplomarbeit. Technische Universität München. Lehrstuhl für Luftfahrtsysteme.

MEIER, M., 2005. *Der Innovations-Prozess.* Skript zur Vorlesung im WS 2005/06 Eidgenössische Technische Hochschule Zürich. Zentrum für Produkt-Entwicklung.

MOET, 2009. *More Open Electrical Technologies: Technical Report (Deliverable D0.02.3)*. URL: http://www.eurtd.com/moet/. [Stand 2011-11-09].

MOF, 2011. *OMG Meta Object Facility (MOF) Core Specification: Version 2.4.1*. URL: http://www.omg.org/spec/MOF/2.4.1/PDF/ [Stand 2011-11-09].

MOHR, B., 2006. *Development and Integration of a combined Air Conditioning - Fuel Cell System for Commercial Aircraft.* Diplomarbeit. Technische Universität München. Lehrstuhl für Luftfahrttechnik.

MOHR, B., 2010. *Neuartige Modularisierungskonzepte für die Rumpf- und Kabinengestaltung: Abschlussbericht - LuFo IV Verbundprojekt KABTEC*. Technische Universität München. Lehrstuhl für Luftfahrtsysteme.

MOHR, B.; PAULUS, D.; BAIER, H. & HORNUNG, M., 2011. Design of a 450-passenger blended wing body aircraft for active control investigations. IMechE Part G: *Journal of Aerospace Engineering*. Accepted for publication on 19. Sep. 2011.

MÖHRINGER, S., 2003. *Die neue Richtlinie VDI 2206: Entwicklungsmethodik für mechatronische Systeme: Motivation, Vorgehen und Ergebnisse der neuen VDI-Richtlinie*. URL: http://mechatronics-net.de/gerworkgroup/sitzung6/moehringer-vdi.pdf [Stand 2011-09-29].

MOIR, I. & SEABRIDGE, A. G., 2003. *Civil avionics systems*. London: Professional Engineering Publ.

MURPHY, L., 2005. Falcon 7X Flies to Market with PLM. *Time Compression Technologies*.

NEGELE, H., 2006. *Systemtechnische Methodik zur ganzheitlichen Modellierung am Beispiel der integrierten Produktentwicklung.* 2. Aufl. München: Utz.

NO MAGIC INC., 2012. *Magicdraw*. URL: https://www.magicdraw.com/files/manuals/MagicDraw%20UserManual.pdf [Stand 2012-01-13].

OESTEREICH, B., 2001. *Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified modeling language.* 5. Aufl. München, Wien: Oldenbourg.

OMG, 2011. *Object Management Group Website*. URL: http://www.omg.org.

PAHL, G. & BEITZ, W., 2007. *Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung ; Methoden und Anwendung.* 7. Aufl. Berlin: Springer.

PARDESSUS, T., 2004. Concurrent engineering development and practices for aircraft design at Airbus*, ICAS- International Council of the Aeronautical Sciences.* Yokohama, Japan.

PICOT, A. & BAUMANN, O., 2007 Modularität in der verteilten Entwicklung komplexer Systeme: Chancen, Grenzen, Implikationen. *Journal für Betriebswirtschaft*, 57, Seiten 221–246.

PONN, J., 2007. *Situative Unterstützung der methodischen Konzeptentwicklung technischer Produkte.* 1. Aufl. München: Hut.

PRATS, M. M. & LAVOIE, C., 2011. Hybrid Fuel Cell-based emergency power architecture for commercial aircraft: *3rd Conference of the European Aerospace Societies (CEAS)*. Venedig, Italien 24. Okt. 2011.

PRATT, M. J., 2001. Introduction to ISO 10303 - the STEP Standard for Product Data Exchange. *J. Comput. Inf. Sci. Eng*(1), 102–103.

PROSTEP IVIP E.V., 2010. *STEP AP242 – Das Automotive/Aerospace Anwendungsprotokoll der Zukunft. Pressemitteilung*. Darmstadt.

QAMAR, A., 2011. *An integrated approach towards model-based mechatronic design*. Stockholm: KTH Royal Institute of Technology. Industrial Engineering and Management.

RANGAN, R. M.; U. A., 2005. Streamlining Product Lifecycle Processes: A Survey of Product Lifecycle Management Implementations, Directions, and Challenges. *J. Comput. Inf. Sci. Eng* 5(3), 227 (11 pages).

RAYMER, D. P., 2006. *Aircraft design: A conceptual approach.* 4. Aufl. Reston, VA: American Institute of Aeronautics and Astronautics (AIAA).

SAAKE, G.; HEUER, A. & SATTLER, K.-U., 2008. *Datenbanken: Konzepte und Sprachen.* 3. Aufl. Heidelberg: Mitp bei Redline.

SABBAGH, K., 1996. *21st century jet: the making and marketing of the Boeing 777*. New York, NY: Scribner.

SASAKI, F., 2004. *Sekundäre Informationsstrukturierung.* Dissertation. Universität Bielefeld. Fakultät für Linguistik und Literaturwissenschaft.

SCHICHTEL, M., 2002. *Produktdatenmodellierung in der Praxis*. München: Hanser.

SCHILLER, F., 2009. *Modellbildung und Simulation: Skript zur Vorlesung SS09.* Technische Universität München. Informationstechnik im Maschinenwesen.

SCHÖNHERR, M., 2004. *Enterprise Architecture Frameworks*. Technische Universität Berlin. Competence Center EAI.

SGO-ITD, 2011. *Systems for Green Operations (SGO): Integrated Technology Demonstrator at a glance. Factsheet*. URL: http://www.cleansky.eu/content/project/system-green-operations. [Stand 2012-01-23].

SHEA, K., 2011. *Computer Aided Product Development: Lecture Script*. Technische Universität München. Lehrstuhl für Produktentwicklung.

SIMPSON, T., 2004. Product platform design and customization: Status and promise. *AIEDAM* 18(01).

SJÖSTEDT, C.-J., 2009. *Modeling and simulation of physical systems in a mechatronic context*. PhD thesis, KTH School of Industrial Engineering and Management. Stockholm.

SPUR, G. & KRAUSE, F.-L., 1997. *Das virtuelle Produkt: Management der CAD-Technik*. München: Hanser.

THE OPEN GROUP, 2006. *The Open Group Architecture Framework (TOGAF) and the US Department of Defense Architecture Framework (DoDAF)*. San Francisco, CA, USA.

TORENBEEK, E., 1982. *Synthesis of subsonic airplane design*. Delft: Delft University Press

ULRICH, K. T. & EPPINGER, S. D., 2008. *Product design and development.* 4. Aufl. Boston: McGraw-Hill Higher Education.

UML, 2011. *OMG Unified Modeling LanguageTM (OMG UML), Infrastructure*. URL: http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/ [Stand 2011-11-10].

VDI, 1993. *VDI 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*. (03.100.40). Berlin: Beuth.

VDI, 2004. *VDI 2206: Entwicklungsmethodik für mechatronische Systeme*. (Bd. 03.100.40; 31.220Bd). Berlin: Beuth.

VDI, 2010. *VDI 3633 Blatt 1 Entwurf: Simulation von Logistik-, Materialfluss- und Produktionssystemen*. Berlin: Beuth.

VERMEULEN, B., 2007. *Knowledge based method for solving complexity in design problems.* Dissertation. Delft University of Technology. Faculty of Aerospace Engineering.

VILSMEIER, J. 2011. *PLM Systeme in der industriellen Praxis: Skript zur Vorlesung*. Technische Universität München. Lehrstuhl für Automatisierung und Informationssysteme.

VOSSEN, G., 2008. *Datenmodelle, Datenbanksprachen und Datenbankmanagement-systeme.* 5., überarb. und erw. Auflage. München, Wien: Oldenbourg.

WALTER, U., 2010. *Systems Engineering. Skript zur Vorlesung SS2010*. München. Technische Universität München. Lehrstuhl für Raumfahrttechnik.

WASMER, A. M., 1998. *Methodische Vorgehensweise beim Entwurf von Mappings zwischen Produktmodellen.* Aachen: Shaker.

WEBER, C.; WERNER, H. & DEUBEL, T., 2002. *A Different View on PDM and its Future Potenzials*. Internation Design Conference - DESIGN 2002. Dubrovnik, Kroatien 14.-17.5.2002.

WEILKIENS, T., 2008. *Systems Engineering mit SysML/UML: Modellierung, Analyse, Design.* 2., Heidelberg: Dpunkt-Verl.

WILDEMANN, H., 2004. *Komplexitätsmanagement: Leitfaden zur Einführung eines durchgängigen Komplexitätsmanagements im Unternehmen.* 5. Aufl. München: TCW-Verlag.

WINTER, E.; MOSENA, R. & ROBERTS, L., 2010. *Gabler Wirtschafts-Lexikon: Die ganze Welt der Wirtschaft: Betriebswirtschaft, Volkswirtschaft, Wirtschaftsrecht, Recht und Steuern*. Wiesbaden: Gabler.

WOHLGEMUTH-SCHÖLLER, E., 1999. *Modulare Produktsysteme*. Frankfurt am Main, New York: P. Lang.

WOLTERS, H., 1995. *Modul- und Systembeschaffung in der Automobilindustrie: Gestaltung der Kooperation zwischen europäischen Hersteller- und Zulieferunternehmen*. Wiesbaden: Dt. Univ.-Verl.; Gabler.

ABKÜRZUNGSVERZEICHNIS

AAA	Advanced Aircraft Analysis
ACARE	Advisory Council for Aeronautics Research in Europe
ACM	Air Cycle Machine
ADM	Architecture Development Method
AEA	All Electric Aircraft
AMDB	Aircraft Model Data Base
ANSI	American National Standards Institute
AP	Application Protocol (im Kontext des Dateiformates STEP)
APP	Aircraft Performance Program
APU	Auxiliary Power Unit
ARP	Aerospace Recommend Practice
ASCII	American National Standard Code for Information Interchange
ATA	Air Transport Association of America
AUTOSAR	AUTomotive Open System ARchitecture
AV	DoDAF All View
BADA	Base of Aircraft Data (EUROCONTROL Flugzeugdatenbank)
CAA	Component Application Architecture
CAD	Computer Aided Design
CADM	Core Architecture Data Model
CAE	Computer Aided Engineering
CATIA	Computer Aided Three-Dimensional Interactive Application
CATVBA	CATIA Visual Basic for Applications
CAx	Computer Aided x
CE	Concurrent Engineering
CFD	Computational Fluid Dynamics
COM	Component Object Model
CPM	Core Product Model
CS	Certification Specifications for Large Aeroplanes

CSV	Comma separated values (Textdateiformat zur Speicherung oder zum Austausch einfach strukturierter Daten)
DBMS	Datenbankmanagementsystem
DCL	Data Control Language
DDB	Verteilte Datenbanken (engl.: distributed databases)
DDL	Data Definition Language
DFS	Detailed Functional Specification
DIN	Deutsches Institut für Normung e. V.
DML	Data Manipulation Language
DMU	Digital Mock Up
DoDAF	Department of Defense Architecture Framework
DXF	Drawing Interchange Format
EAI	Enterprise Architecture Integration
ECS	Environmental Control System
ED	EUROCAE Document
EDMS	Engineering Data Management System
eECS	Elektrisches Environmental Control System
EIA	Electronic Industries Alliance
EIS	Entry into Service
EMF	Eclipse Modeling Framework
EMV	Elektromagnetische Verträglichkeit
EN	Europäische Norm
EPS	Electrical Power System
ERM	Entity-Relationship-Model
ETL	Extraktion - Transformation - Laden
EU	Europäische Union
EUROCAE	European Organization for Civil Aviation Equipment
FEM	Finite-Elemente-Methode
FUS	Fuel Systems
GUI	Graphical User Interface
HLS	High Lift Systems
HMI	Human Machine Interface
HPS	Hydraulic Power Systems
IEC	International Electrotechnical Commission
IFE	In-Flight Entertainment
IGES	Initial Graphics Exchange Specification

IMA	Integrierte Modulare Avionik
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
IT	Informationstechnologie
ITD	Integrated Technology Demonstrator
ITP	Instruction to proceed
JAR	Joint Aviation Authorities
JDBC	Java Database Connectivity
JT	Jupiter Tessellation
LGS	Landing Gear Systems
MBS	Multi Body Simulation
MBSE	Modellbasiertes Systems Engineering
MEA	More Electric Aircraft
MODAF	Ministry of Defence (UK) Architecture Framework
MOET	More Open Electrical Technologies
MOF	Meta Object Facility
MOSES	Modellbasierte Systementwicklung in der Automobilindustrie
MS	Microsoft
MySQL	MY Structured Query Language (Verwaltungssystem für relationale Datenbanken)
NAF	NATO Architecture Framework
NAI	Nacelle Anti Ice
NIST	National Institute of Standards and Technology
ODBC	Open Database Connectivity
OEM	Original Equipment Manufacturer
OMG	Object Management Group
OODB	Objektorientierte Datenbank
ORDB	Objekt-relationale Datenbank
OV	DoDAF Operational View
PAX	Passengers
PDM	Produktdatenmanagement
PFCS	Primary Flight Control Systems
PID	Proportional– Integral– Derivativ Kontrollglied einer Regelstrecke
PLM	Product Lifecycle Management
POA	Power Optimized Aircraft
PPS	Pneumatic Power Systems

PrADO	Preliminary Aircraft Design and Optimisation (Flugzeugentwurfsprogramm TU Braunschweig)
Pro/E	Pro/ENGINEER Wildfire
QG	Quality Gate
QVT	Query View Transformation
RDB	Relationale Datenbank
RTCA	Radio Technical Commission for Aeronautics
SAE	Society of Automotive Engineers
SCADE	Safety-Critical Application Development Environment
SE	Systems Engineering
SGO	Systems for Green Operations
SQL	Structured Query Language
SQLJ	Eingebettetes SQL für Java
SRA	Strategic Research Agenda
SRD	System Requirement Document
STEP	Standard for the Exchange of Product Model Data
STL	Surface Tessellation Language
SV	DoDAF Systems View
SysML	Systems Modeling Language
TLARD	Top Level Aircraft Requirement Document
TOGAF	The Open Group Architecture Framework
TRD	Test Requirements Document
TV	DoDAF Technical Standards View
UML	Unified Modeling Language
UPDM	Unified Profile for DoDAF and MODAF
VBA	Visual Basic for Applications
VDI	Verband Deutscher Ingenieure e.V.
VPM	Virtual Product Management
VRML	Virtual Reality Modeling Language
WIPS	Wing Ice Protection System
XML	Extensible Markup Language
XMLDB	XML Datenbank
XSLT	Extensible Stylesheet Language Transformation

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ANNEX I – LUFTFAHRTSPEZIFISCHE SOFTWARE

Tabelle A1 zeigt einen Ausschnitt der gängigen Softwarewerkzeuge, die in der Entwicklung von Flugzeugen und Flugzeugsystemen Anwendung finden. Die Tabelle erhebt keinen Anspruch auf Vollständigkeit, deckt aber große Teile der Werkzeuglandschaft in Forschung und Industrie ab.

Tabelle A1 – Softwarewerkzeuge in der Entwicklung von Flugzeugsystemen

	Tool	Hersteller	Dateiformat	Verschlüsselung	Export aller Modelldaten	Export in offene Formate
Aircraft Design	PrADO	TUBS/ IFL	*.dat	offen	ja	*.dat
	AAA / APP	DarCorporation	*.analysis	proprietär	ja	*.txt
	Pacelab Suite	PACE	*.plkdeoconcept, …	proprietär	nein / eingeschränkt[1]	*.step, *.xml, …
	ODIP	Airbus FPO	*.sc, …	offen	ja	
	Piano	Lissys Ltd.	*.dat, …	offen	ja	
CAD	CATIA	Dassault Systemes	*.catpart *.catproduct	proprietär	nein / eingeschränkt[2]	*.iges, *.step
	ProEngineer	PTC	*.prt, *.asm	proprietär	nein / eingeschränkt[3]	*.iges, *.step
	SolidWorks	Dassault Systemes	*.sldprt, *.sldasm	proprietär	nein / eingeschränkt[3]	*.iges, *.step
	AutoCAD	Autodesk	*.dwg	proprietär	nein / eingeschränkt[3]	*.dxf
MBS	ADAMS	MSC Software	*.bin, *.cmd, *.out, …	proprietär	ja	*.iges, *.step, …
FEM	ANSYS	Ansys	*.db, *.out	proprietär	nein / eingeschränkt[3]	*.iges, *.step, …
	NASTRAN	MSC Software	*.nas, *.dat, *.bdf	proprietär	nein / eingeschränkt[3]	XDB, MNF
	PATRAN	MSC Software	*.db, …	proprietär	nein / eingeschränkt3	*.iges, *.xmt, *.step
	MSC Laminate	MSC Software	*.layup	proprietär	nein / eingeschränkt	*.txt
	Modeler	Ansys	*.fedb, …	proprietär	nein / eingeschränkt	*.iges, *.step, …
CFD	FLUENT	Ansys	*.cas, *.dat, *.cdat, …	proprietär	nein / eingeschränkt	verschiedene
	Star CCM	CD Adapco	*.ccm	proprietär	nein / eingeschränkt	ICEM
Technical Computing	MATLAB	Mathworks	*.m, …	proprietär	ja	*.txt
	SciLab	Digiteo / INRIA	*.sc, …	proprietär	ja	*.txt
	Excel	Microsoft	*.xls	proprietär	ja	*.csv
	Mathcad	PTC	*.mcd, *.xmcd	proprietär	ja	*.xml, *.pdf, *.html
	Mathematica	Wolfram	*.wdx, *.cdf	proprietär	ja	*.xml, …
	Maple	Maplesoft	*.mpl	proprietär	ja	*.txt, …
	Simulink	Mathworks	*.mdl	proprietär	nein / eingeschränkt[3]	*.xls, …
	XCOS (Scilab)	Digiteo / INRIA	*.xcos	proprietär	nein / eingeschränkt[3]	*.xml
	Modelio	Modeliosoft	*.ofpx	proprietär	ja	*.xmi
	Dymola	Dassault Systemes	*.m	proprietär	ja	*.txt

[1] Export nur teilweise möglich, Einbindung von DLLs
[2] Export über neutrale Datenformate bzw. Parameterexport über VB & CATVBS
[3] Export über neutrale Datenformate

ANNEX II – DIGITAL MOCKUP

Abbildung A1 zeigt das digitale Modell eines Verkehrsflugzeuges in der Größe des Airbus A320-200. Das Modell dient der geometrischen Integrationsanalyse von Systemen wie dem Leitungssystem der Klimaanlage. Über Programmierschnittstellen wird der Datenaustausch des DMU und der Metadatenbank realisiert und damit die Kommunikation von Geometrieparametern, Messungen oder Kollisionsanalysen an andere Domänen innerhalb der Systementwicklung ermöglicht. Die beispielhafte Gestaltung von Schnittstellen innerhalb der in dieser Arbeit verwendeten CAD-Software CATIA V5 ist in Annex V zu finden.

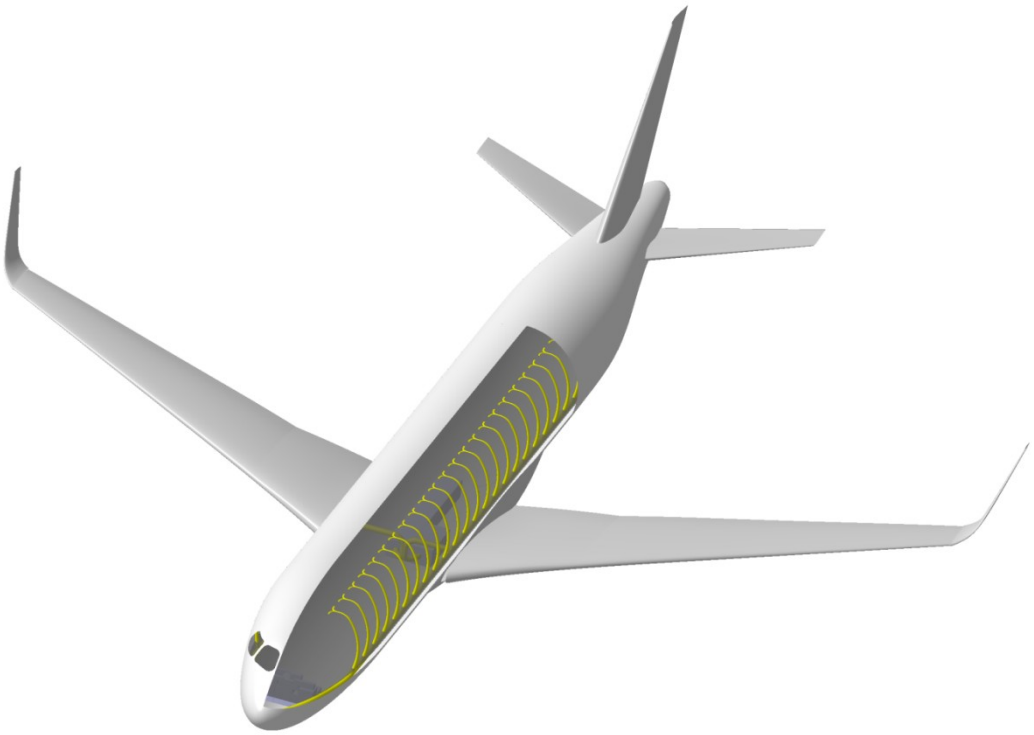


Abbildung A1 - Isometrische Ansicht des DMU & ECS-Leitungen

ANNEX III – MODELLBASIERTE INFRASTRUKTUR

Abbildung A2 zeigt die modellbasierte Infrastruktur, die im Rahmen dieser Arbeit geschaffen wurde. Alle am Entwurf beteiligten Domänen kommunizieren domänenübergreifende Datensätze über das Metamodell eines Systems, welches durch eine relationale Datenbank realisiert wird. Die Kommunikation der Partialmodelle und der Systemdatenbank wird durch die anwendungsabhängige Schnittstellengestaltung ermöglicht, wobei integrierte und benutzerdefinierte Schnittstellen zum Einsatz kommen können (Annex IV & Annex V). Die Dateiverwaltung über ein PDM-System, welche die im Entwicklungsverlauf entstehenden Versionen der Partialmodelle mit den Versionen des Metamodells koppeln, wurde nicht realisiert.

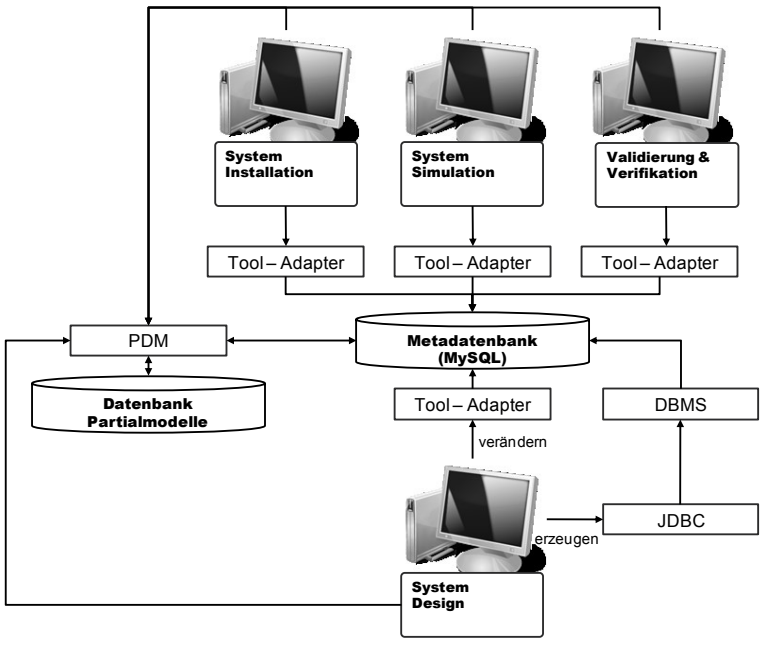


Abbildung A2 – Beispiel zur modellbasierten Infrastruktur

ANNEX IV – INTEGRIERTE SCHNITTSTELLEN

Ein Großteil der Softwarewerkzeuge, die in der Entwicklung von Luftfahrtsystemen Anwendung findet, beinhaltet eine integrierte Schnittstelle zur Anbindung von Datenbanken und der Kommunikation von Datensätzen über diese Schnittstelle mittels der Semantik der SQL. Am Beispiel der Software Matlab soll die Verwendung bereits vorhandener Datenbankschnittstellen demonstriert werden. Matlab stellt die Schnittstelle JDBC zur Verfügung, welche die Anbindung verschiedener Datenbanken unterstützt, unter anderem auch die für diese Arbeit benützte MySQL-Datenbank.

Für den Aufbau einer Verbindung `conn` zur Datenbank `aircraftData` werden Benutzerinformationen und der Standort der Datenbank (`localhost:3306`) an den Befehl `database` übergeben.

```
conn = database('aircraftData','root','',...
'com.mysql.jdbc.Driver',...
'jdbc:mysql://localhost:3306/aircraftData');
```

Code A1 - Aufbau einer Datenbankverbindung in Matlab

Nach der Erstellung einer Verbindung zur Datenbank können Daten von der Datenbank in den Arbeitsspeicher der aktuellen Anwendung über eine SQL-Anfrage geladen werden. Der folgende Befehl lädt beispielsweise alle Daten (`*`) eines Record, in dem das Attribut `ID` mit dem String `A320` belegt ist und schreibt sie in das Array `curs`. Der Befehl `assignin` weist schließlich den dritten Platzhalter im Array `curs` als Wert zur Variable `manufacturer` zu:

```
sqlquery = ['select * from aircraftType where ID = "'A320'"'];
curs = fetch(exec(conn, sqlquery));
assignin('base','manufacturer',cell2mat(curs.Data(3)))
```

Code A2 - Erzeugung einer Matlab- Variable aus einem RDB- Datensatz

Weiterhin können Variablen des Arbeitsspeichers (`header, ndata`) von Matlab in die Tabelle `aircraftType` der Datenbank geschrieben werden.

```
mktab = 'CREATE TABLE aircraftType (ID char(50),mName char(50))';
exec(conn, mktab);
colnames = {header{1}, header{2}};
exdata = {ndata(1),ndata(2)};
fastinsert(conn, 'aircraftType', colnames, exdata)
```

Code A3 - Erzeugung eines RDB- Datensatz aus Matlab- Variablen

Diese stark strukturierte Vorgehensweise ist in allen Anwendungen mit bereits integrierter Schnittstellenfunktionalität über JDBC (bzw. SQLJ, ODBC) möglich und erlaubt es jedem Anwender bereits nach kurzer Einarbeitungszeit die Erstellung von Skripten zur bidirektionalen Kommunikation von Softwareanwendung und externen Datenbanken.

ANNEX V – BENUTZERDEFINIERTE SCHNITTSTELLEN

Im Gegensatz zu Software mit integrierten Datenbankschnittstellen, müssen Datensätze aus Softwarewerkzeugen ohne diese Schnittstellen über externe, benutzerdefinierte Schnittstellen ausgetauscht werden. Dazu wird über ein benutzerdefiniertes Programm auf die Anwendung direkt zugegriffen oder das zugehörige, proprietäre Datenformat manipuliert. Am Beispiel der CAD-Software CATIA V5 wird nachfolgend die Gestaltung von benutzerdefinierten Schnittstellen beschrieben. Über die internen Sprachelemente der *Component Application Architecture* (CAA) kann auf jedes beliebige Objekt und seine Attribute innerhalb von CATIA V5 zugegriffen werden. Die CAA entspricht der Architektur eines *Component Object Model* (COM), welches die Manipulation von CAA- Objekten unabhängig vom Betriebssystem oder der Programmiersprache zulässt. Über die Dokumentation der CATIA-CAA kann die Struktur aller DMU-Elemente erfasst und für die Schnittstellengestaltung herangezogen werden. Abbildung A3 zeigt beispielhaft die Automatisierungsobjekte eines Teiledokumentes.

Im Rahmen der exemplarischen Methodenanwendung dieser Arbeit war die automatisierte Untersuchung von Strukturinteraktionen des ECS innerhalb des DMU notwendig. Dazu wird ein VBA-Skript (`CATMain()`) ausgeführt, welches zunächst das DMU (`mockup.CATProduct`) im Stapelverarbeitungsmodus lädt und eine Verbindung zur Systemdatenbank aufbaut. Über die Verbindung `oConn` wird das Parameterset `myRecSet` aus der Systemdatenbank (`aircraft_system_data`) ausgelesen und eine entsprechende Parametervariationen innerhalb des Bauteils `ducts` vorgenommen. Für jedes Parameterset wird eine Kollisionsanalyse durchführt und deren Ergebnisse werden sowohl in der Systemdatenbank als auch in einem XML-Bericht (`clash.xml`) gespeichert. Der zugehörige Quellcode ist in Code A4 wiedergegeben, was dem Leser dieser Arbeit die eigene Implementierung von Schnittstellen zwischen CATIA V5 und einer relationalen Datenbank erleichtern mag.

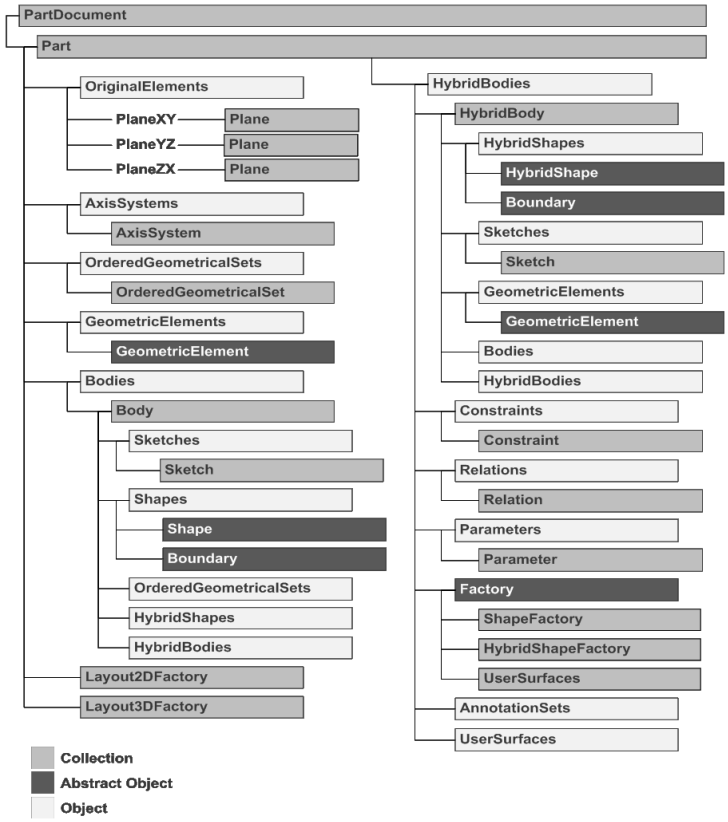


Abbildung A3 – CATIA-CAA PartDocument Automatisierungsobjekte

```
Sub CATMain()
'*************************************************************
'open catproduct
'*************************************************************

 Set documents1 = CATIA.Documents

 Set productDocument1 = documents1.
      Open("C:\Simulationen\SysID\clash\mockup.CATProduct")

  '*************************************************************
'connect to system database
'*************************************************************
Dim oConn As ADODB.Connection ' Connection Handle
Set oConn = New ADODB.Connection
oConn.Open "DRIVER={MySQL ODBC 5.1 Driver};" & _
                    "SERVER=127.0.0.1;" & _
                    "DATABASE=aircraft_system_data;" & _
                    "USER=root;" & _
                    "PASSWORD=;" & _
                    "Option=3"

'*************************************************************
'read parameters to be analyzed from system database
'*************************************************************

Dim myRecSet As New ADODB.Recordset 'Recordset Object
Dim strSQL1 As String ' String variable to store sql command
strSQL1 = "SELECT * FROM 083_systems_pipes"
myRecSet.Open strSQL1, oConn, adOpenKeyset
```

```
'************************************************************
'loop until last parameter set (end of file) is reached
'************************************************************

Dim i As Integer 'Runtime variable

i = 1

While Not myRecSet.EOF

       Set partDocument1 = documents1.Item("ducts.CATPart")
       Set part1 = partDocument1.PartSet parameters1 = part1.Parameters
       Set Length1 = parameters1.Item("Main_pipe_diameter")
       Length1.Value = myRecSet(3) 'Third variable
       part1.Update

   '************************************************************
   'detect clashes between "ducts" and "frames" for current parameter set
   '************************************************************

       Set products1 = productDocument1.Product.Products
       Dim FirstProd As Product
       Dim SecondProd As Product
       Set FirstProd = products1.Item("ducts.1")
       Set SecondProd = products1.Item("frames.1")

       'create groups for clash computation
       Dim objGroups As Groups
       Set objGroups = CATIA.ActiveDocument.Product.
                           GetTechnologicalObject("Groups")
       Dim grpFirst As Group
       Dim grpSecond As Group
       Set grpFirst = objGroups.Add()
       Set grpSecond = objGroups.Add()

       'add selected products to groups
       grpFirst.AddExplicit FirstProd
       grpSecond.AddExplicit SecondProd
       Set cClashes = CATIA.ActiveDocument.Product.
                           GetTechnologicalObject("Clashes")
       Set oClash = cClashes.AddFromSel

       'set new clash to be computed between two groups (two selected products)
       oClash.FirstGroup = grpFirst
       oClash.SecondGroup = grpSecond
       oClash.ComputationType = catClashComputationTypeBetweenTwo
       oClash.Compute

'************************************************************
'generate DB output values (default: 2/clearance)
'************************************************************

Flag = 2
Comment = "Clearance : ConflictStatusIrrelevant"
Set cConflicts = oClash.Conflicts
Dim j As Integer ' Runtime variable

For j = 1 To cConflicts.Count
       Set oConflict = cConflicts.Item(j)
       If (oConflict.Type = catConflictTypeClash) Then

       oConflict.Status = catConflictStatusRelevant
              oConflict.Comment = "Clash : ConflictStatusRelevant"

    ElseIf (oConflict.Type = catConflictTypeContact) Then

       oConflict.Status = catConflictStatusIrrelevant
              oConflict.Comment = "Contact : ConflictStatusIrrelevant"

    End If

    Flag = oConflict.Status
       Comment = oConflict.Comment

Next
```

```
'************************************************************
'write clashes to database
'************************************************************

Dim strSQL2 As String ' String variable sql command 1
Dim strSQL3 As String ' String variable sql command 2
Dim myRecSet2 As New ADODB.Recordset 'Recordset Object
strSQL2 = "UPDATE 086_systems SET Flag_SysID ='" & Flag & "' WHERE
             ID='Sys_ECS_00" & i & "'"
strSQL3 = "UPDATE 086_systems SET Comment_SysID ='" & Comment & "' WHERE
             ID='Sys_ECS_00" & i & "'"
Set myRecSet2 = New ADODB.Recordset
myRecSet2.Open strSQL2, oConn, adOpenDynamic, adLockOptimistic
myRecSet2.Open strSQL3, oConn, adOpenDynamic, adLockOptimistic

'************************************************************
'export clashes to clash.xml in C:\Simulationen\SysID\clash\XML
'************************************************************

oClash.Export ClashSpecExportAsXMLClashSpec,
      "C:\Simulationen\SysID\clash\XML_ClashReport_" & i & "\clash" & i & ".xml"

'************************************************************
'move the RecordSet pointer to the next parameter position
'************************************************************

myRecSet.MoveNext
i = i + 1
Wend

oConn.Close

End Sub
```

Code A4 - Export einer Kollisionsanalyse durch ein CATVBA-Skript