On physical aspects of cabin architectures using tolerancing methods

A thesis accepted by the Faculty of Aerospace Engineering and Geodesy of the Universität Stuttgart in partial fulfillment of the requirements for the degree of Doctor of Engineering Sciences (Dr.-Ing.)

by
Dipl.-Ing. Benjamin Landes-Dallat
born in Stuttgart-Bad Cannstatt, Germany

Main referee:	Priv.Doz Dr.-Ing. Stephan Rudolph
Co-referee:	Prof. Dipl.-Ing. Rudolph Voit-Nitschmann
Co-referee:	Prof.dr.ir. Michel van Tooren
Date of defense:	May 6, 2013

Institute of Statics and Dynamics of Aerospace Structures
University of Stuttgart
2013

Acknowledgments

The first thanks go to Dr. Stephan Rudolph from the University of Stuttgart. He encouraged me to use an innovative method and software frame set I could build my work on. I always could come with any question and our numerous discussions made me advance. The working atmosphere and the team spirit in the research group were extraordinarily good. I am very glad that Airbus continues to use this wide research experience. I appreciated very much that Professor Voit-Nitschmann accepted me as doctoral student. He opened the door to the University of Stuttgart for me coming from the industrial context, for which I am very thankful. I also feel very honored that Professor van Tooren with his experience in the field of knowledge-based engineering methods acted as third examiner.

A very special thank you goes to Mr. Paul Clozel from the École Centrale de Lyon and to Pierre-Alain Rance from MECAmaster. I gratefully received information and help concerning MECAmaster far beyond a regular scope. The students from Lyon were a very helpful support.

Within the Airbus community at Hamburg, a special thanks goes to Dr. Arndt Stephan, who initiated everything. He set up a research project tailored for me and also had me working in an industrial tolerancing project, which turned out to be a very fruitful combination for the success of this work. The discussions with my colleague and friend Kenneth Boenning, leader of the latter project, were a continuous source of ideas – also beyond this dissertation. My team leader, Stefan Lee, walked with me the entire way from start to end, outstandingly supporting and pushing me wherever possible. There were a variety of colleagues and students, who answered to my questions and who supported. I cannot name all of them here, but their help was very much appreciated.

I am grateful to my parents for their love and their support. Thank you for going with me when I chose my ways. My Mom being author also acted as lector and my Dad being designer and graphics designer helped me with the design of the figures. Thanks also go to them and to my Dad's wife Evi for hosting and feeding me when I came from Hamburg to Stuttgart to go to the University. Otherwise, this 'multi-city project' would have been associated with much less familiarity. Here, I also need to thank my brother Christopher who helps me to keep learning beyond engineering sciences.

I am very glad that my friends Lucas Ogden and Neil Chisholm are native English speakers, providing me well appreciated help as lectors. Without them, this dissertation would be written in immature English. I am responsible for any remaining language mistakes.

At the end, I would like to thank Chang-Mi. Our love is the most important thing in my life. You always accompany me on my paths, you show me your perspective and you make my life more complete I ever thought it could be. Together we grow and live. I love you.

Contents

Abbreviations and Symbols

Abbreviation	**Meaning *(proper names in italics)***
1D, 3D	one-dimensional, three-dimensional
3DCS	*3-Dimensional Control System*
AAP	architecture analysis parameter
ADCATS	*Association for the Development of Computer-Aided Tolerancing Systems*
AIRG	air grid
AKC	assembly key characteristic
ASME	*American Society of Mechanical Engineers*
ATA	*Air Transport Association of America*
CAD	computer-aided design
CAE	computer-aided engineering
CAM	computer-aided manufacturing
CAQ	computer-aided quality assurance
CAT	computer-aided tolerancing
CATIA	*Computer-Aided Three-Dimensional Interactive Application*
CFD	computational fluid dynamics
CDL	cabin design languages
CDL_LR	cabin design languages for a long range aircraft
CLNG	ceiling
CM	cabin module
CSS	central substructure
DCC	*Design Computing and Cognition*
DCS	*Dimensional Control Systems*
DFSS	design for six sigma
DFMA	design for manufacture and assembly
DMU	digital mock-up
DOF	degree of freedom
EN	*European Standards*
F2F	floor-to-floor
FAL	final assembly line
FE	finite element
FEM	finite element method
GAIA	*Graphical Analysis of Interfaces for Assemblies*
GASAP	geometry as soon as possible
GD&T	geometric dimensioning and tolerancing
GPS	geometric product specification
GUI	graphical user interface
ISO	*International Organization for Standardization*

Abbreviation	**Meaning *(proper names in italics)***
KBE	knowledge-based engineering
KC	key characteristic
LHS	left-hand side
MBE	model-based engineering
MCA	major component assembly
MDA	model-driven architecture
METUS	*Management Engineering Tool for Unified Systems*
Mh	man-hour
MKC	manufacturing key characteristic
MON	monument
OHSC	overhead stowage compartment
PDM	product data management
PKC	performance key characteristic
PLM	product life cycle management
PrADO	*Preliminary Aircraft Design and Optimization Program*
PreSTo	*Preliminary Sizing Tool*
RBE	requirement-based engineering
RHS	right-hand side
RSS	root square sum *(precisely: square root of the sum of the squares)*
SWL	sidewall lining
SysML	*Systems Modeling Language*
UML	*Unified Modeling Language*

Symbol	**Meaning**
α	angle of displacement
c	influence coefficient within 3D tolerance calculations
$c_{j,i}$	influence coefficient of tolerance contributor t_i on assembly tolerance T_j
c_{lin}	linearized influence coefficient
c_{exact}	exact influence coefficient
D	geometrical dimension
i	index for tolerance contributors
IT	tolerance grade
j	index for assembly tolerances
l	length
n	number of tolerance contributors
m	number of assembly tolerances
t	tolerance contributor
t_i	tolerance contributor i
T	assembly tolerance
T_G	assembly tolerance of a gap
T_j	assembly tolerance j
T_O	assembly tolerance of an offset
T_{RSS}	assembly tolerance calculated with statistical method
$T_{1.5xRSS}$	assembly tolerance calculated with safety-factored statistical method
T_{wc}	assembly tolerance calculated with worst-case method

Abstract

In the conceptual design phase for the development of aircraft cabins the question about product architectures, spatial, mechanical and functional interfaces as well as the integration into the fuselage play a central role. While *functional* aspects are in the foreground for the conceptual design of cabin systems like the air distribution system or data systems, the architectures of the cabin modules like the stowage bins and the entire lining panels of the passenger compartment are mainly characterized by interdependencies between *physical* aspects, in particular between the geometrical shape, mechanical, functional and operational behavior as well as manufacturing aspects.

The physical interfaces between the cabin modules, the fuselage structures and the attachment brackets are of high relevancy. *Tolerance management*, which defines the repercussions of tolerances already in the early conceptual phase of product development, here accomplishes important tasks. It can provide the required interconnection between geometrical shape, manufacturing-related deviation from nominal size, mechanical-functional behavior and the wide field of repercussions on manufacturing – in particular on the final assembly line.

The analysis of physical aspects of cabin architectures consequently becomes a problem exceeding pure geometrical considerations. For this reason, methods based on a geometry paradigm for the generation and analysis of product data considering only geometrical aspects reach their limits concerning their validity for physical architecture aspects. On the other hand, the consideration of additional product data models in parallel is time-consuming. In particular, if several technical scenarios need to be compared, analysis methods like tolerance calculations are often omitted, since the relation between the modeling time and the validity of the calculation results based on values coming from heuristic or synthetic estimation procedures seems to be too unfavorable.

In contrast, modern model-based methods, such as for instance, graph-based design languages enable interdisciplinary product models which are customized exactly to the needs of the respective problem. In addition to this, approaches with design languages comprising design rules offer the possibility for a fast and reproducible generation and modification of product data models. In the context of this dissertation so-called *cabin design languages* are developed that can describe and model *physical aspects of cabin architectures including tolerancing*. Key aspects of these design languages are the concepts of ‘physical components’ and ‘physical interfaces’ along with the associated aspects for physical integration like tolerances and installation processes.

The implementation of these design languages consists of an extensive cabin-specific class diagram and a set of graph-based rules which together allow generating and calculating multiple variants of a technical scenario. The classes and rules also comprise synthetic estimations for component tolerances or masses, for example. The software-based implementation additionally provides routines which transform the compiled cabin models into analysis and visualization models. Amongst other, this comprises automatized means for the preparation of tolerance analyses, the calculation of analysis parameters in terms of ‘metrics’, the conceptual representation of manufacturing processes or the exchange of product data models.

A use case demonstrates the practical benefit of executable design languages for the named problem. A cabin segment including the corresponding design rules is modeled. By means of parametrical and topological changes, several technical scenarios including the corresponding analysis and visualization models can be generated and evaluated. The following discussion examines the applicability of the presented method for industrial praxis.

It shows, that conceptual tolerance management in conjunction with further analysis methods and supported by design languages can play a primary role for the industrial evaluation of physical aspects of cabin architectures in the conceptual design phase.

Kurzfassung

In der Konzeptphase der Entwicklung von Flugzeugkabinen spielen die Fragen nach Produktarchitektur, nach räumlichen, mechanischen und funktionalen Schnittstellen und nach der Integration in den Rumpf eine zentrale Rolle. Während bei der Konzeptionierung von Kabinensystemen wie der Luftversorgung oder den Datensystemen *funktionale* Aspekte im Vordergrund stehen, ist die Architektur der Kabinenmodule wie die der Gepäckfächer und der gesamten Verkleidungselemente des Passagierraumes hauptsächlich gezeichnet von wechselseitiger Abhängigkeit *physischer* Gesichtspunkte, insbesondere zwischen geometrischer Gestalt, mechanischem, funktionalem und operationellem Verhalten sowie Fertigungsaspekten.

Von großer Bedeutung sind die physischen Schnittstellen zwischen Kabinenmodulen, der Rumpfstruktur und den Halterelementen. Das *Toleranzmanagement*, welches die Auswirkungen von Toleranzen schon in der frühen Konzeptphase der Produktentwicklung festlegt, übernimmt hierbei wichtige Aufgaben. Damit kann die nötige Verbindung zwischen geometrischer Gestalt, den fertigungsbedingten Abweichungen von Nominalmaßen, mechanisch-funktionalem Verhalten und den weitreichenden Auswirkungen auf die Fertigung – speziell auf die Endmontagelinie – gezogen werden.

Die Analyse physischer Architekturaspekte von Flugzeugkabinen wird damit zu einer Problemstellung, die über rein geometrische Betrachtungen hinausgehen muss. Aus diesem Grund stoßen auch Methoden basierend auf einem Geometrieparadigma zur Generierung und Analyse von Produktdaten, die nur geometrische Aspekte berücksichtigen, an ihre Grenzen bezüglich der Aussagekraft über physische Architekturaspekte. Die parallele Betrachtung zusätzlicher Produktdatenmodelle dagegen ist zeitaufwendig. Insbesondere wenn mehrere technische Szenarien verglichen werden müssen, wird auf Analyseverfahren wie Toleranzberechnungen oftmals verzichtet, da das Verhältnis zwischen der Zeit für das Modellieren und der Aussagekraft von Rechenergebnissen basierend auf Werten gewonnen aus heuristischen oder synthetischen Schätzverfahren zu ungünstig erscheint.

Moderne modellbasierte Methoden, wie beispielsweise graphenbasierte Entwurfsprachen, ermöglichen dagegen interdisziplinäre Produktmodelle, die genau an die Bedürfnisse der jeweiligen Fragestellung angepasst sind. Zusätzlich dazu bieten Ansätze mit Entwurfsprachen durch die Abbildung der Entwurfsregeln die Möglichkeit, Datenmodelle schnell und wiederholbar zu generieren und zu modifizieren. Im Rahmen dieser Dissertation werden sogenannte *Kabinenentwurfssprachen* (engl. *cabin design languages*) entwickelt, welche *physische Aspekte von Kabinenarchitekturen inklusive Toleranzmethoden* beschreiben und abbilden können. Zentrale Aspekte dieser Entwurfsprachen sind die Begriffe „physische Komponente“ und „physische Schnittstelle“ mit den damit verbundenen physischen Integrationsaspekten wie Toleranzen oder Installationsabläufe.

Die Implementierung dieser Entwurfsprachen beinhaltet ein umfangreiches kabinenspezifisches Klassendiagramm und ein graphenbasiertes Regelwerk, welche gemeinsam ermöglichen, verschiedene Varianten eines technischen Szenarios zu erzeugen und zu berechnen. Die Klassen und Regeln schließen auch synthetische Abschätzgleichungen, zum Beispiel für Bauteiltoleranzen oder Komponentengewichte, mit ein. Die softwarebasierte Umsetzung sieht zusätzlich Routinen vor, welche fertig kompilierte Kabinenmodelle in Analyse- und Visualisierungsmodelle transformieren. Dies beinhaltet unter anderem auto-

matisierte Methoden zur Vorbereitung von Toleranzanalysen, zur Berechnung von Analyseparametern im Sinne von „Metriken“, zur konzeptionellen Darstellung der Fertigungsabläufe oder zum Austausch von Produktdatenmodellen.

Ein konkreter Anwendungsfall demonstriert den praktischen Nutzen ausführbarer Entwurfsprachen für die benannte Problemstellung. Dabei wird ein Kabinensegment mitsamt den dazugehörenden Entwurfsregeln modelliert. Mithilfe parametrischer und topologischer Veränderungen können mehrere technische Szenarien inklusive der entsprechenden Analyse- und Visualisierungsmodelle erzeugt und ausgewertet werden. Mit der anschließenden Diskussion wird die Anwendbarkeit der vorgestellten Methodik in der industriellen Praxis betrachtet.

Es zeigt sich, dass konzeptionelles Toleranzmanagement in Verbindung mit weiteren Analysemethoden durch die Unterstützung von Entwurfsprachen eine tragende Rolle für die industrielle Bewertung physischer Aspekte von Kabinenarchitekturen in der Konzeptphase spielen kann.

Chapter 1

Introduction

1.1 Aircraft Cabin

For many aircraft operators the *cabin* is the place to put a unique branding to differentiate them from other airlines. For airline customers – the passengers – the *cabin* is a place where they have to remain, sometimes hours, using various cabin functions. From the airplane manufacturers' point of view, the *cabin* comprises these two aspects – and many more, looking at the entire life cycle [40, 117]. On one hand, the 'visible' cabin components like lining panels, overhead bins, galleys, lavatories and stowages need to be designed so that it is possible to adapt them to the wishes of airliners, taking operational aspects including use, maintenance and repair into consideration. On the other hand, all safety and passenger comfort-related aspects need to be developed and integrated, including systems which are primarily 'invisible' for the passenger like those for cabin intercommunication, air distribution and emergency oxygen along with hundreds of meters of wires and ducts. As a whole, the product *aircraft cabin* needs to fulfill all functional requirements and industrial demands such as short development and reduced manufacturing lead time, thus improving cost effectiveness.

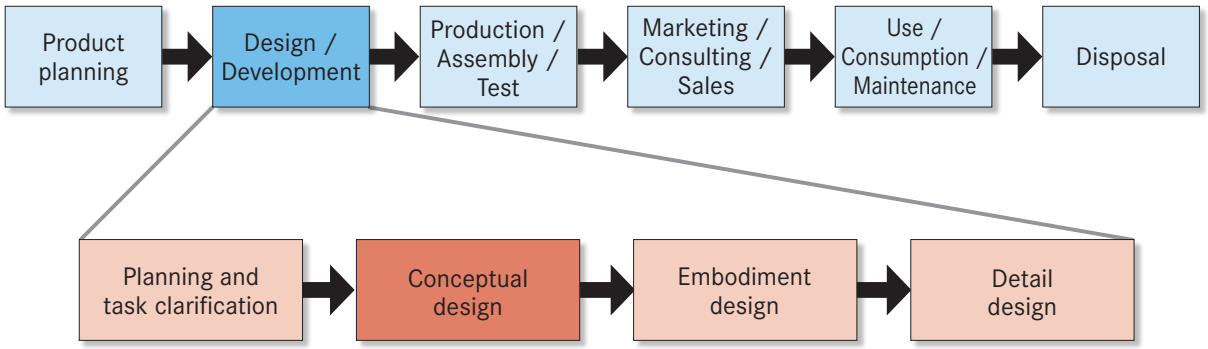

Figure 1.1: *The product life cycle phases in general and the product development phases in detail according to* PAHL *and* BEITZ *[117]*

Similar to other products, the life cycle of an aircraft cabin can be broken down into phases such as conceptual design and architecture, design phase, design verification phase along with the manufacturing phase as well as the full spectrum of deployment, operation, maintenance and disposal according to systems engineering terminology[1] [40]. PAHL and BEITZ use similar definitions and cluster product development phases into 'planning and task clarification', 'conceptual design', 'embodiment design' and 'detail design' [117] (fig. 1.1). The focus of this dissertation lies on the development of virtual means for improvements of the conceptual design phase[2].

[1] Systems engineering refers to a frame set of methods for design and development processes in the context of the whole product life cycle [40, 117]. Various methods such as customer-focused requirement-based engineering (RBE), requirement validation, function-driven design, verification and systems engineering management are linked together to give structure to multi-disciplinary design processes and to support complex products. Among other engineering disciplines, the aerospace sector benefits from systems engineering methods, for example in the context of design languages (see section 1.3) or model-based engineering (MBE) approaches (see section 1.2), while software development processes make use thereof as well [167].

[2] The picture shown refers to the definitions by PAHL and BEITZ [117]. It represents a very generic development process and does not imply that the process is purely sequential. With modern development processes and planning methods like systems engineering and concurrent engineering [31, 40] used for aircraft development, the individual steps can be adjusted for optimizations. There can be iterative loops between or within the named phases. Moreover, by making use of digital design methods, the evolving critical paths can be reduced or new critical paths can emerge since they are pending on the execution length of the individual steps [57, 58, 90, 134, 136] (see also subsection 1.1.3 and sections 1.2 and 1.3).

1.1.1 Cabin Architecture and Integration

The product architecture [30] of complex products like an aircraft or its highly integrated components like the cabin and the related systems has a manifold influence on the product development process. In this context, describing architectures goes beyond a pure geometrical decomposition. Integrated product development for instance includes manufacturing-related aspects [45] as well as functional descriptions [124] and design constraints from multiple domains [43].

Aircraft cabin architectures have to be substantial concepts that allow integrating the entire cabin including all modules and systems into the fuselage by keeping the entire life cycle of the complete aircraft in mind. To maintain an overview about the multi-disciplinary problem of architecture definition and the related integration problems, it can be helpful to use 'parameters' [121, 127, 129, 159] or 'metrics' [43] during the design process in order to enable qualitative architecture evaluation.

The aspects of systems engineering [31, 40, 117] and of know-how reuse [134] or knowledge-based engineering (KBE [163]) also can come into consideration at this stage, offering a wide range of means to support architecture development and analysis.

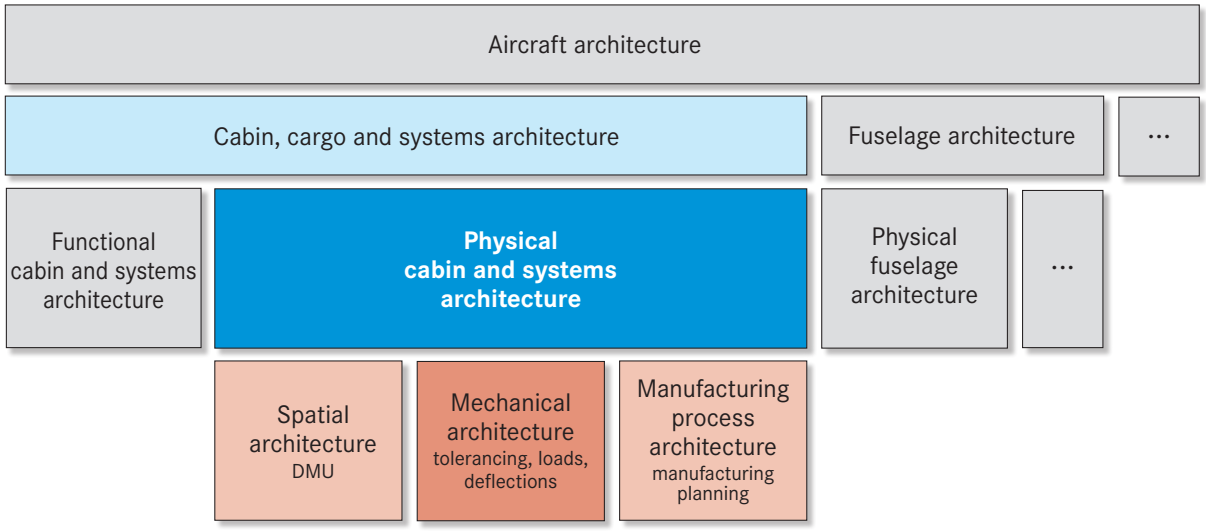

Figure 1.2: *Facets of cabin architecture work*

It is possible to distinguish between different facets of architectural work, as figure 1.2 shows. *Functional cabin architecture* deals with the functional definition of all systems and system components that are integrated into other cabin components or directly into the fuselage. It also has a look at the implementation of the systems into the physical product and the resulting functional and operational dependencies [107].

Physical cabin architecture compromises firstly the conceptual physical definition of modules or components fulfilling the required cabin functions[3] along with the corresponding space allocation for the integration of the modules into the fuselage. This includes the accessibility for installation, maintenance and repair, but also the consideration of tolerances for functionality, quality and installation as well as of deflections due to in-flight movement. Secondly, it contains the mechanical aspects of the attachment interfaces of cabin modules and system components including all brackets and corresponding functional tolerances, masses, loads and deflections. Thirdly, it involves considering installation processes in the

[3]The traditional allocation of functions on modules or components is described by the 'ATA chapter breakdown' [3] as discussed in the subsequent subsection.

final assembly line (FAL) and the linking to the major component assembly (MCA) of the fuselage segments and thus industrialization and cost aspects.

Of course, functional and physical facets are strongly interconnected and cannot be considered independently. As the title of this dissertation suggests, this work primarily represents a contribution on the physical aspects of cabin architectures[4].

1.1.2 ATA Chapter Breakdown

Traditionally, the aerospace sector uses the 'ATA chapter breakdown' [3] to describe all components of an aircraft in a standardized way throughout the whole documentation[5]. This comprises a unique functional allocation of functions and aircraft systems to ATA chapters.

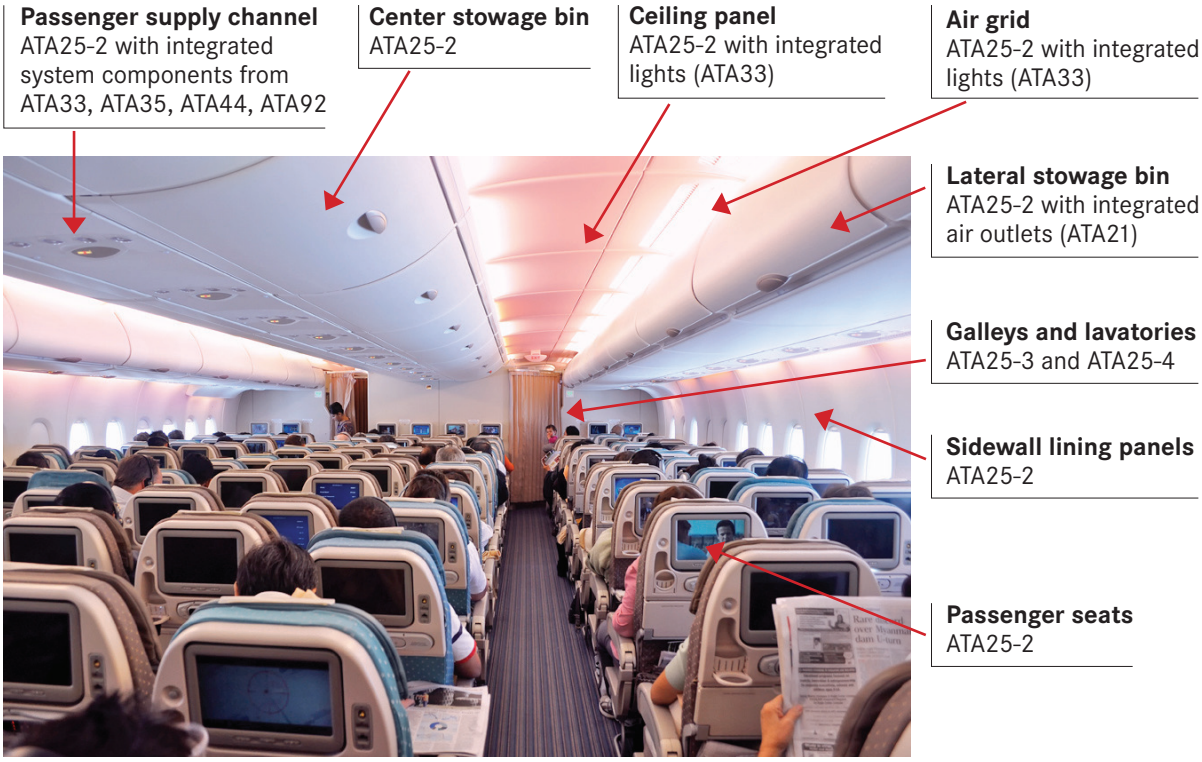

Figure 1.3: *Some cabin modules and the related ATA chapters*[6]

ATA53 describes the fuselage structure, for instance. The systems are documented for example within ATA33 (lights), ATA35 (oxygen), ATA38 (water/waste), ATA44 (cabin intercommunication) and ATA50 (equipment and systems for cargo). ATA25 contains all furnishing and equipment components, and is subdivided into ATA25-2 for the cabin compartment consisting of the lining components (ceiling and sidewall panels, door lining, door frame lining, entrance area lining as well as special customized cabin installations), the overhead stowage bins and the passenger supply channel hosting the integrated reading lights, call buttons, oxygen mask dispensers and individual air nozzles. ATA25-3 refers to the galleys, ATA25-4 to the lavatories, just to name a few (see fig. 1.3).

[4] Links to functional aspects will be outlined in subsection 6.2.2.

[5] See also http://www.s-techent.com/ATA100.htm for a detailed listing, accessed Jan 2012.

[6] Courtesy of EUGENE YEO, see http://www.airliners.net, photo ID: 2004016, accessed Jan 2012.

Even though system components usually require a focus on functional design and architecture at first, many aspects of cabin architectures are of physical nature. Due to the diverse physical interfaces of cabin modules and components with the fuselage, many technical boundary conditions are predefined for physical cabin design and architectures by the fuselage architecture. For example, the frame distance of the fuselage along with the stringers plays an important role for cabin integration, since almost any cabin and system attachment brackets need to be fixed to them. Consequently, most lining modules or the stowage bins within the passenger compartment have length variants, which fit exactly between the frame bays.

1.1.3 Architecture Philosophies with a Focus on Interfaces

Industrial experience tells that the attachment brackets - and therefore the physical interface between cabin and fuselage including its functional requirements - are taken care for only in a limited way during the early conceptual design phase. Mostly, only load-related aspects are considered, but hardly tolerance-related aspects with all their industrial implications. The disadvantage of this working process for cabin architectures is, that crucial fuselage decisions may be taken without the contribution and the consideration of cabin-related architecture, design and industrialization aspects. Additionally, the ATA paradigm with a strict separation of functions allocated to decoupled physical systems limits a more integrated functional way of thinking towards new design concepts. In particular, there are no ATA chapters for non-physical design aspects with multi-domain interdependencies, such as 'gaps or split lines between components', 'mechanical interface functions' or 'installation processes'.

Nowadays, in the international business environment, the need for cost reduction comes along with the demand for leaner and faster production with stable product quality. Consequently, the focus moves towards concurrent engineering and systems engineering principles [40], where multiple components need to be developed in parallel to shorten the development phase [31, 90, 134]. GÖPFERT [57, 58] talks of 'modularization' of physical products and product components as a method to reduce uncertainty during system design. In this reading, modularization strategies are considered as auxiliary means for designers to simplify designing and integration. Uncertainties such as for instance complexity, innovation, dynamics or unclear design purpose or target definitions mean, that the system designer has a lack of knowledge about system elements or their interrelation. PAHL and BEITZ also understand product architecture as *'[...] a scheme showing the relationship between the function structure of a product and its physical configuration [...]'* and move it into the context of modularization strategies[7] [117].

However, these frame conditions lead to highly complex products and development processes, where many multi-domain design and product aspects link and influence each other. In 2008, the NEW YORK TIMES published a newspaper article with the headline *'Parts didn't click together [...]'* [11]. It states how an airplane manufacturer *'[...] has ended up with a pile of parts and wires, and lots of questions about how they all fit together [...]'*. In a different context, KRIEGEL describes a very intuitive example, where at the beginning of the development process for an office copier the systematic consideration of interface aspects has not taken place [89]. Thus the interfaces design has been 'improved' several times until an over-constraint mechanism with tight tolerance requirements evolved, which failed at last. The author shows, that a focus on the interface's *function* during the design phase gives the chance to solve the problem pragmatically and make it even cheaper for manufacturing.

It is obvious that it is necessary to set a focus on the *physical interfaces* between the modules or subsystems constituting an assembly [122, 162]. HILLSTÖM investigates the interfaces between modules and where to locate these [70]. He proposes a concept to identify optimized interface locations by embedding axiomatic design principles [99, 157, 158], tolerancing methods and design for manufacturing and

[7] Compare also with the functional decomposition approach by PIMMLER and EPPINGER [122].

assembly (DFMA) methods. The evolving research will be discussed a bit more in detail in section 2.4. PRICE et al. discuss the interface behavior of complex engineering systems and show, that mechanical interface management is linked to functional descriptions [124].

In order to describe assembly processes, SCARR [138] talks of geometry features called 'functional surfaces' and clusters them according to their *functional behavior* into 'provides support', 'transmits forces', 'locates the component in the assembly', 'provides location for other components in the assembly' and 'transmits motion (bearing surfaces)'. In contrast to SCARR, WHITNEY [170] proposes mapping the interface function on abstract model aspects called 'assembly joints' rather than on geometry features. He differentiates between so-called 'mate' and 'contact' joints. The first one comprises an assembly step, where two parts are geometrically located relative to each other and certain degrees of freedom are constrained between corresponding geometry features. The latter one plays only a supportive role in the assembly like an additional nut fixing two parts together after they already were located relative to each other using mate joints.

MANTRIPRAGADA incorporates the concept of joints into a classification proposal for different assembly types [104]. A so-called 'type-I assembly' or 'part-defined assemblies' means that all constraints to locate the component are defined by the geometry features themselves. During assembly, the parts can be put together without any need for adjustment or location moves with respect to other features. In practical tolerancing experience, such parts are called 'positively located parts'. Consequently, 'type-II assemblies' are installation processes for which the final location of a component is not implicitly predefined by its geometry, but where additional assembly joints leading to adjustment steps are necessary. MARGUET et al. take the principle of interface functions and bring it into the context of tolerancing using assembly analysis [106].

Also for the development of conceptual aircraft architectures the role of manufacturing and industrialization aspects are more and more emphasized. KRAUSE et al. link modularization with innovative aircraft cabin integration scenarios focusing on manufacturing-related aspects [88]. ATA25 modules can be clustered in a more functional way to optimize assembly processes using DFMA methods [61]. Overall, modern design or development methods offer the chance to rethink traditional development processes and technical solutions for future cabin architectures [88].

Altogether it becomes clear that methods for conceptual interface design including functional tolerancing and assembly aspects are 'key players' for cabin architectures in terms of physical integration. These methods offer to link conceptual physical design, quality and manufacturing for architectural considerations.

1.2 Current Means for Physical Integration

There is a wide field of software methods supporting the efforts of conceptual design, architecture analysis and technical interface management. This section can only provide an insight looking at well-known engineering methods along with some methods from the field of research. All contributions to this field related to tolerancing methods are presented later in chapter 2.

1.2.1 Spreadsheet-Based Methods

Spreadsheet-based methods can be used to collect data, to process data such as making data table calculations, to exchange data and to visualize data in tables or charts. One standard software for these applications is EXCEL[8].

In particular for conceptual work, spreadsheet proves to be a simple, lean and effective way for both scientific and industrial research [2] and conceptual engineering. For instance, the aircraft weight books

[8] See http://office.microsoft.com/en-us/excel/, accessed Jan 2012.

can be listed and controlled in spreadsheets or product data lists such as tolerance lists[9] can be maintained and exchanged. In a certain way, this expresses the fact, that in modern, computer-oriented world, manual calculation and simple data handling is still an important mean to manage or even solve engineering problems – pending on the volume of the research questions.

To extend the broad bandwidth of functions and application possibilities, macros can be implemented in spreadsheets for (semi-)automatic data processing[10].

1.2.2 Computer-Aided Design and Associated Methods

Of course, computer-aided design (CAD) plays an important role in the context of conceptual engineering for physical integration. There are many purposes of representing geometrical product data within digital mock-ups (DMU) during the concept phase of design, such as clash detection, checks for readiness or completeness of design data, spatial investigations for ergonomics, human factors, manufacturability and maintainability as well as spatial interface definitions.

There are many different CAD tools in industrial use, which can be used for the above tasks supporting the user with graphical user interfaces (GUI). Within the context of this work, CATIA V5[11] has been used, due to the fact that this tool is in wide industrial use within the aerospace sector.

Modern CAD systems provide a wide portfolio of geometry-based design and analysis methods to support computer-aided engineering (CAE) [9] linked with geometrical design data. Alternatively, external software or integrated methods often access the 3D data representation or derived data models as a basis for data model analysis methods. Among many others, examples for such CAE methods are:

- Finite element methods (FEM) for load and stress analysis use 3D geometry data as a basis for the generation of finite element (FE) meshes [85].
- The same applies to geometry preprocessing for computational fluid dynamic (CFD) methods [16].
- Computer-aided manufacturing (CAM) approaches can use the product data tree and the geometry data to model assembly processes [10, 14, 130].
- Methods for computer-aided tolerancing (CAT) constitute one central topic of this dissertation and will be described in detail in chapter 2.
- Integrated CAD applications within frameworks for preliminary aircraft design methods, for KBE and for graph-based design languages will each be outlined below.

CAD and associated methods are usually based on a ‘geometry paradigm’ [16, 134]. This means that the model structure consists of geometry elements and describes geometrical or physical entities. It is not foreseen to implement design aspects beyond physics or geometry independently from geometrical objects. Workaround solutions are possible, where virtual geometry components are created to represent more abstract product data, literally acting as a ‘carrier bag’ for non-geometrical data or parameters.

1.2.3 Preliminary Aircraft Design Methods

Traditional preliminary aircraft design methods like those put forth by RAYMER [127], ROSKAM [129] or TORENBEEK [159] use combinations of empirical, semi-analytical and analytical formulas and correlations to make iterative estimations for design parameters or metrics. This implies the generation and use of databases to feed the estimations with interpolations and extrapolations [121]. Nowadays, there is a

[9] See section 2.6 and subsection 5.1.5.

[10] E.g., within subsection 5.1.6 spreadsheet-based macros are used to visualize installation process charts.

[11] See http://www.3ds.com/de/products/catia, accessed Jan 2012.

tendency towards computer-based methods to support iterative, multi-disciplinary design and to speed up or automatize processes for modeling and designing, as the tools PrADO [66] or PreSTo [2] demonstrate.

LEDERMANN [96, 97] proposes an associative-parametric method for preliminary aircraft structure design. The hierarchical model structure is based on CAD data and offers parametric links between different dynamic objects. Weight, mass, cost and aerodynamic analyses can be conducted. The method has is applied and extended for preliminary weight estimations for preliminary fuselage and wing design [83, 116, 168].

The work presented by ARMSTRONG, MAWHINNEY et al. [9, 110, 111] focuses on the implementation of simplified, parameterized geometry handling for the interaction between CAD and CAE. ARMSTRONG presents 3D modeling requirements for different applications, proposes feature-based design methods and derives a simulation model for preliminary aircraft design purposes [9]. MAWHINNEY introduces a geometry-based approach to analysis integration for conceptual aircraft design [111] using analysis-driven design principles [110], for which an integration framework for conceptual aircraft design based on systems engineering principles has been developed [44, 109, 125]. Various geometry-based design disciplines are linked benefiting from the aforementioned interactions between CAD and CAE.

VAN TOOREN et al. put focus on the research about KBE methods [163] for preliminary aircraft design [92, 161]. The concept foresees so-called 'high-level primitives', which are knowledge data objects [146]. They support a fast compilation of aircraft variants using a knowledge-based 'multi model generator' for multi-disciplinary design analysis and optimization [91, 145].

1.2.4 Industrialization and Cost Aspects

The noted methods for DFMA [61, 88] and for assembly modeling [104, 170] or CAM [10, 14] stand for the necessity to link designing with questions concerning industrialization and cost. CURRAN [32] and WATSON [166] represent contributions to link preliminary aircraft design methods with a parameter-based aircraft design cost model. PAHL and BEITZ [117] as well as EHRLENSPIEL [46] show more in general, that modern engineering needs to consider cost-related aspects.

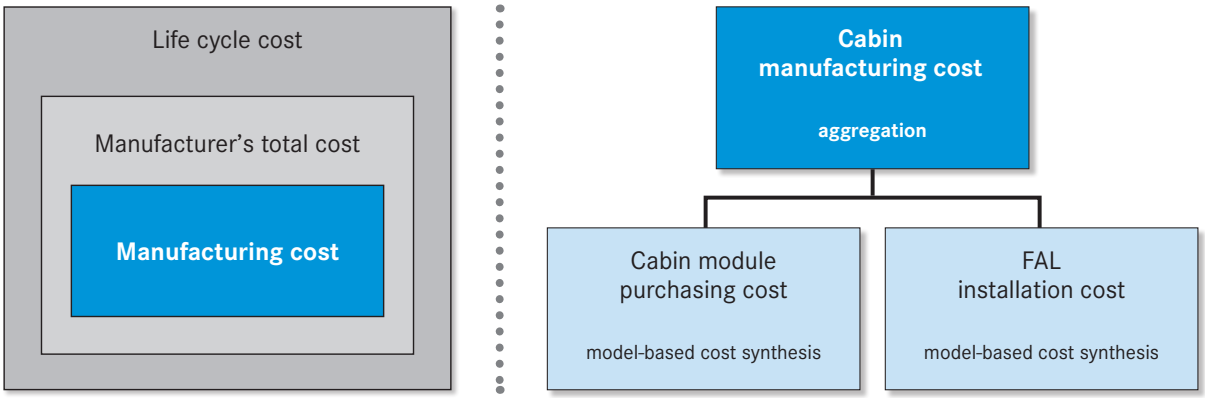

Figure 1.4: *Cost classification according to* EHRLENSPIEL *[46] (left side) and potential breakdown of cabin manufacturing cost according to* PAHL *and* BEITZ *[117] (right side)*

For this purpose, EHRLENSPIEL proposes a concept which distinguishes between 'manufacturing cost', the 'total cost' for the manufacturer and 'life cycle cost'[12] [46], as the left-hand side of figure 1.4

[12]The corresponding original German terms according to EHRLENSPIEL are 'Herstellkosten', 'Selbstkosten' and 'Lebenslaufkosten' [46] (translations by the author of this dissertation).

shows. The first one comprises all product-related material and manufacturing costs. The total cost is the sum of the manufacturing cost and other cost, which cannot be assigned directly to the product, such as administration or overhead cost, for instance. The life cycle cost summarizes all costs from development, production and use. For physical integration and architecture analysis tasks, the cost of primary interest is the recurring manufacturing cost [117], which can be decomposed in a simplified way as sum of the purchasing cost for the cabin modules and its subcomponents and of the cost for the installation efforts in the FAL (right-hand side of fig. 1.4). The mentioned literature can be used as reference for cost and price prediction methods as well. Section 2.5 also mentions some cost synthesis methods related to tolerance synthesis and quality assessment.

1.2.5 Model-Based Engineering

Model-driven architectures (MDA) come from software design as approach to handle software complexity [60, 136]. Flexible and adaptable data models[13] are used to represent and group aspects of software. POOLE [123] describes visions, standards and emerging technologies from the field of MDA which make use of object-oriented programming means [128], such as the differentiation between the class and instance model level (fig. 1.5).

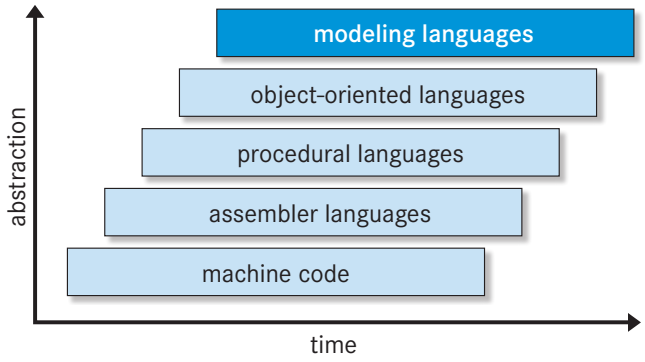

Figure 1.5: *Modeling languages supporting model-based engineering approaches (illustration according to* REICHWEIN *[128] and* RUDOLPH *[136])*

Model-based approaches for software engineering problems conduct model transformations [33] to get from one model to another. It is possible to classify the source-target relationship of these transformations into rules which update, extend, delete or replace the data source by a target. Transformations, which generate code or text as target are called 'model-to-code transformations' or 'model-to-text transformations'. If both the source and the target are models, the transformation in between is called 'model-to-model transformation'. The source and the target model objects do not stringently need to be instances of the same class model. Among various others, the *Unified Modeling Language* (UML) can be used for formal model representation [15, 128]. The *Systems Modeling Language* (SysML) basically is a subset of the UML[14] with special extensions for systems engineering purposes [167].

[13] STACHOWIAK [155] presents a wide historical background for modeling or model-building, and also provides a frame set for modeling approaches in general. He states that models follow three attributes, which are the *attribute of reproduction*, the *attribute of abbreviation* and the *attribute of pragmatical purpose* (translations by the author of this dissertation). In order to understand a model, it is not enough to be able to answer the question *'of* what *is the model reproduced'*. Since every model serves a purpose, answers to the questions *'for* whom*,* when *and* for which purpose *in relation to its pragmatical purpose is the model made'* are required, too.

[14] See http://www.omg.org/spec/UML/ and http://www.omg.org/spec/SysML/ for specifications, accessed Jan 2012.

Nowadays there is the tendency to make use of software modeling methods to build and transform engineering data models. Such approaches are called model-based engineering (MBE) [7, 136, 160]. This knowledge transfer opens a wide field of applications for modern engineering within all product development phases [160]. One major advantage is the possibility to overcome the geometry paradigm [16, 134] and exchange it with a more abstract and more flexible model-based paradigm that can bring 'designing' closer to the modern understanding of multi-disciplinary architecture. Of course, geometry-related CAE aspects are implemented within model structures for physical architectures.

The application possibilities of the MBE paradigm are manifold. PEAK et al. show examples for mechanical system models and simulation in SysML [119, 120], ABULAWI demonstrates the usage of SysML to support CAD modeling for preliminary aircraft design [1] in the context of the already mentioned tool PreSTo [2]. MEHLITZ discusses how aerospace system models in UML can be checked and verified with Java routines [115]. As ARNOLD and RUDOLPH [10] as well as BERGHOLZ [14] show, one tendency of CAM also goes towards model-based design methods. LAUSCHER et al. [95] use SysML to support technical specifications, in this case for railway engineering. Requirement-based engineering (RBE) methods like the 'requirement modeling framework' from eclipse[15] or DOORS[16] open the field to requirement and function specifications. In particular for aircraft systems-related architecture aspects, suitable requirement preprocessing and functional approaches can form an important part of architecture design and analysis [107]. Commercial simulation modeling software like MATLAB Simulink[17] for system simulation can be of help for such applications. The entire context of UML-based design languages will be discussed in the subsequent section 1.3.

CAD data and design or product data in general is often embedded in product data management (PDM) and product life cycle management (PLM) systems [47] with the intention to standardize product data and to make it available in a consistent manner. Where possible, CAE methods such as CAD or CAM access such product databases and benefit from model-based aspects. However, PDM and PLM systems aim at the detailed design aspects during the product development phase rather than on the creation of concept phase models. In the context of functional preprocessing methods for geometry-based models, the commercial software tool METUS (Management Engineering Tool for Unified Systems) can be named. It visualizes the overlapping of the technical product structure and corresponding organizational structures [57, 58]. The tool is capable of separating the physical and functional product structures. It can map the organizational structure accordingly to the engineering object in order to optimize development processes around a specific development project.

Also KBE approaches [163] can follow model-based approaches. For instance, the reuse of knowledge is a key aspect of design languages [134]. RUDOLF copes with KBE for assembly planning in the 'digital factory' using the automotive sector as example [130]. The KBE methods for preliminary aircraft design from VAN TOOREN et al. have already been mentioned. Some further KBE contributions for particular tolerancing-related applications will be named in section 2.4.

[15] See http://eclipse.org/rmf/, accessed Feb 2012.
[16] See http://www.ibm.com, accessed Jan 2012.
[17] See http://www.mathworks.de/products/simulink/index.html, accessed Jan 2012.

1.3 Design Languages

Modern approaches for designing making use of design languages simulate conceptual design processes. ANTONSSON and CAGAN provide a good overview on design languages, distinguishing three different types [8]:

- *String-based design languages*, e.g., L-systems as used by LINDENMEYER [8]
- *Shape-based design languages*, e.g., 3D shape grammars as used by HEISSERMAN [67, 68], CHAU [24] or CAGAN [8]
- *Graph-based design languages*, e.g., grammar-based design by SCHMIDT and CAGAN [141, 142] or graph-based design languages by RUDOLPH [90, 134]

In general, engineers or designers use the concept of 'design rules' with a dedicated 'purpose', which achieve a certain 'effect' on the design object [99]. There is much research about the correlation between design processes and human cognition. For instance, the mentioned conference series DCC [56] as well as GEDENRYD [53] express this field of investigations. In this context, design languages are means for computer-supported engineering design with the intention to automatize formal steps of designing and enabling simple reuse of know-how [8, 134]. For all of the mentioned design language approaches, software implementations have been developed accordingly [8, 24, 67, 90, 134, 142].

Graph-based design languages according to RUDOLPH share a certain level of abstraction regarding the design objects, which are clustered in a 'meta language' [134] or 'class diagram' [128] for syntactic definitions of the corresponding classes or the 'vocabulary' [136] (see fig. 1.6). The substantives of the respective engineering domain are the candidates for the classes, which can have attributes describing the characteristics of the classified object.

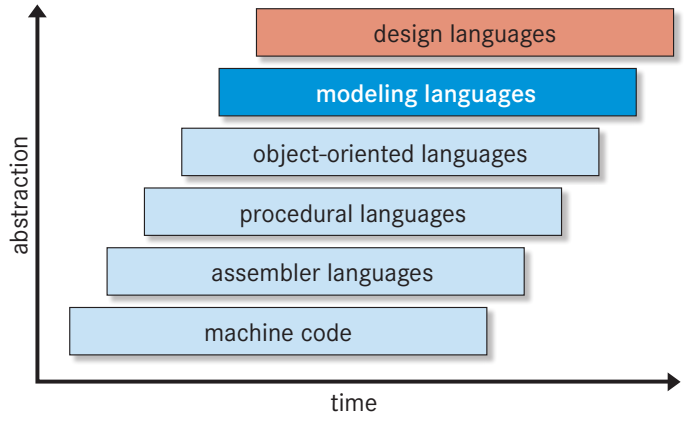

Figure 1.6: *Graph-based design languages extending the model-based paradigm (illustration according to* RUDOLPH *[136])*

In comparison to convenient model-based approaches, graph-based design languages additionally enrich data models with 'graph-based rules' [90], constraint processing techniques [136] and analysis extensions for engineering purposes [75], as figures 1.6 and 1.7 show. The vocabulary along with the design rules constitute a so-called 'semantic hull' [135], for which syntactic correctness can be guaranteed and validated.

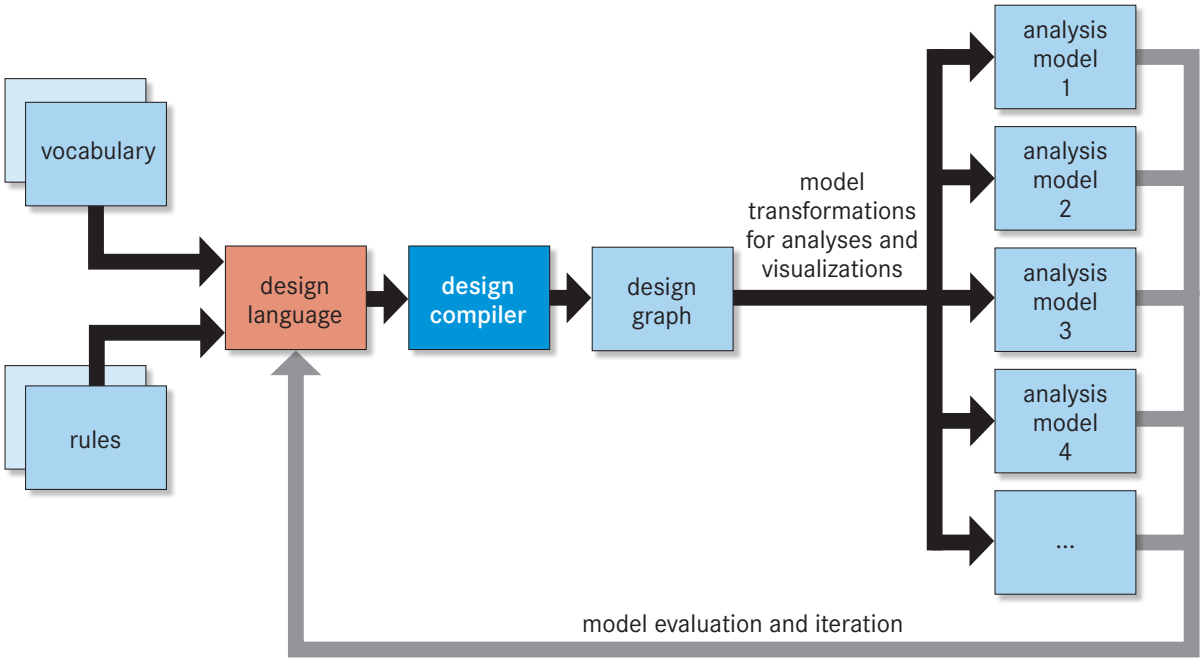

Figure 1.7: *Generic working process of design languages according to* RUDOLPH *[136]*

The rules[18] represent model transformations to build up data models incrementally. Within the left-hand side (LHS) of a rule, objects or object 'patterns' [134] of the design graph are traced. The right-hand side (RHS) specifies how to modify the design graph. The model transformations can comprise parameter changes of existing objects and their attributes, the deletion and replacement of existing objects or the generation of all-new objects.

The kind and sequence[19] of the rules is considered a modeled copy of the real design process. Decision nodes within the rule sequence allow to develop the design according to predefined or iterative parameters. Abstractions on class level allow generalizing rules in order to make them repeatable for multiple applications. Different rules and rule sequences may lead to the same or similar models: the 'effects' can be achieved with different 'rules' [99]. A so-called 'design graph' [90] – the graphical representation of the design data – evolves during the execution of the design rules[20].

After the execution of the design rules, the design models need to be interpreted in the pragmatic context [135], as figure 1.7 sketches. At first, this can be accomplished by comparing different modeling results manually. Alternatively, further model transformations can convert the design graph data into more convenient data formats, which are more 'readable' or 'interpretable' for design engineers. These can be, for example, CAD models [16], spreadsheet visualizations, formal mathematical models or simulation models [128]. By now, various examples for case-specific software model interfaces are available as well, such as for automotive design [62], integrated satellite design [59, 63, 64, 139], integrated airplane design [15, 16], FEM analyses [85, 86, 87], design and development for exhaust aftertreatment systems [164], digital factory models [10] or initial work on aircraft cabin tolerance analysis [93].

[18] Refer to fig. 1.9 for an example of a design rule visualized by an UML object diagram [75].

[19] Refer to fig. 1.8 for an example of a design rule sequence visualized by an UML activity diagram [75].

[20] Fig. 1.10 shows an example for the design graph before and after the execution of a design rule. Since the design graphs shown later within this work get more and more complex (see figures 5.26, 5.30, 5.39 and 5.44), a simplified representation of the UML instances has been selected instead of conventional UML visualization formats. This simplified visualization scheme is already applied in fig. 1.10: each bullet within the graph represents a UML instance, the interconnecting lines correspond to data interfaces between the UML instances.

RUDOLPH embeds design languages in the context of similarity mechanics [133] and sets the basis for continuous research on graph-based design languages following a model-based paradigm [136]. He uses the free development platform eclipse[21] for rule-based compilation of multi-domain models with the software DesignCompiler 43 [75] to implement the generic working process as shown in fig. 1.7.

The modeling language UML is used as the representation format of these design objects [15, 16, 128]. Therefore software plugins are developed to provide specific functions like a rule editor, activity diagrams, graph visualizations or constraint processing techniques [75, 133, 136]. These functions enable the user to tailor-made the design languages including the vocabulary, the rules, the rule sequence and the model transformation to engineering analysis models. Remarkable results already demonstrated the applicability for various design problems [5, 63, 139].

Due to the flexible modular plugin architecture, it is possible to extend the functionalities of design languages by further problem-specific software plugins. For instance, the plugins can be used to perform model-to-model transformations from the design language models (saved in UML format) into software-specific data models. The research work presented by this thesis makes use of the concept, as it will be demonstrated in chapter 5.

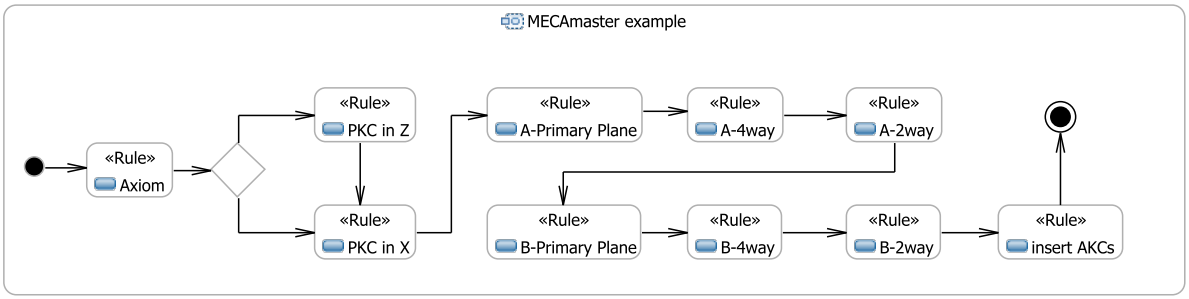

Figure 1.8: *Design language rule sequence including a start node (small black dot), an end node (big black dot), rules (boxes) and a decision node (rhombus)*

Fig. 1.8 shows a rule sequence visualized by an UML activity diagram. The rule sequence creates a model of an assembly consisting of three components[22]. With the first rule ('Axiom'), the three components are added to the model, followed by a decision node (the rhombus in fig. 1.8) deciding about how many target tolerances are added to the model. The decision is pending on a parameter which has been set within the first rule.

The rule 'PKC in Z' adds a target tolerance called 'PKCZ' to the model, which is shown in figures 1.9 and 1.10. Within the rule, two UML objects representing two components of the assembly are searched for. Once found, additional UML objects are created and linked, representing physical edges of the two components as well as a gap in between[23]. The subsequent rules enrich the model with tolerancing data.

Once the rules shown in figure 1.8 are executed, the resulting UML data model can be transformed into tolerance analysis models making use of software plugins developed in the context of this work[24]. Due to the parametric and modular setup of this design language, multiple parametric and topological variants could be compiled and calculated within seconds.

[21] See http://www.eclipse.org, accessed Jan 2012.

[22] The tolerancing-related data of the model will be described in detail in chapter 2. Refer to fig. 2.1, page 20, for an first impression of the three components represented in the design language model.

[23] Although a gap is no physical object, it can be modeled and processed by design languages. This phenomenon shows the flexibility of design languages compared to CAD models following a geometry paradigm as discussed in subsection 1.2.2

[24] See subsection 5.1.1

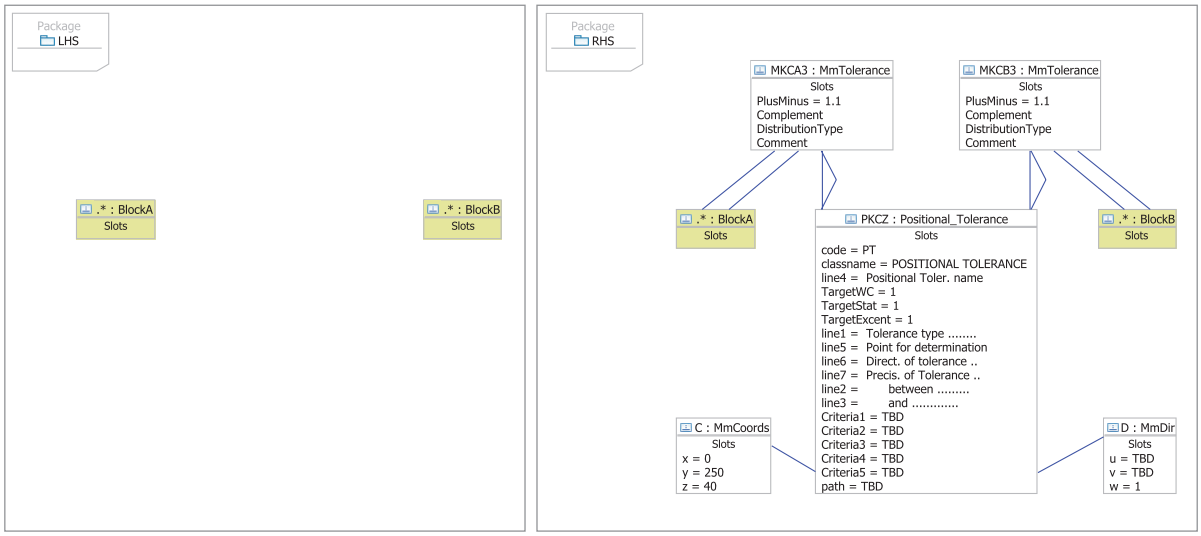

Figure 1.9: *Rule 'PKC in Z'. In the LHS two UML objects (representing two components of an assembly) are searched for, in the RHS new objects are created and linked to them.*

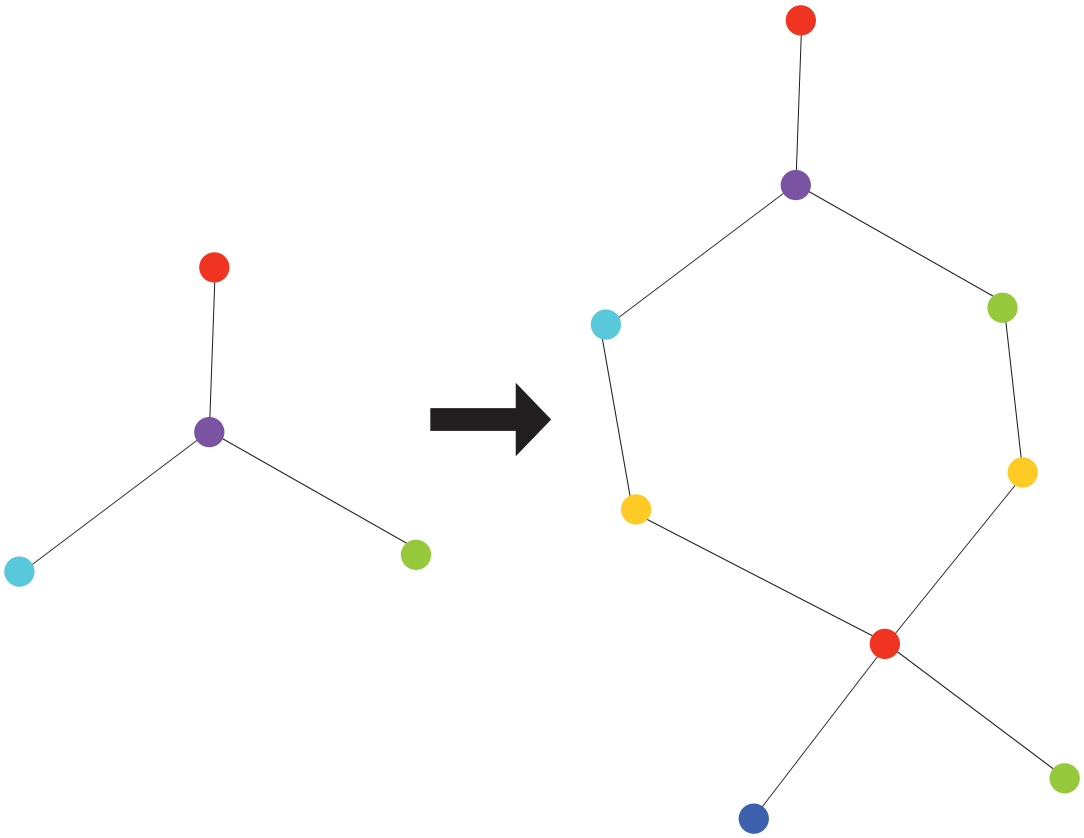

Figure 1.10: *Abstract visualization of the design graph* before *and* after *the execution of the rule 'PKC in Z' as shown in fig. 1.9. Each bullet represents a UML instance, the interconnecting lines correspond to data interfaces between the UML instances. Instance '.*:BlockA' from the LHS of fig. 1.9 corresponds to the light blue bullet, '.*:BlockB' corresponds to the green one, 'PKCZ:Positional_Tolerance' from the RHS corresponds to the lower red one on the RHS.*

Chapter 2

Tolerancing Methods

2.1 Background and Terminology

Any measurable geometrical feature faces deviations from its specified nominal geometry. This imperfectness cannot be omitted and production with 'zero deviation' is not possible. To cope with this phenomenon, tolerances are specified to allow for reasonable geometrical variation [82]. The manifold implications of tolerances can be actively designed and controlled during product development.

'Deviation' (ISO 286-1:2010) describes the actual difference between the specified nominal dimension and the actual, measured dimension. As every geometrical feature of every manufactured part deviates from its specified nominal geometrical size, position and orientation, it is necessary to specify a permissible variation – which is a 'tolerance' – for this feature and its attributes so that associated functions can be ensured. Accordingly, ISO 286-1:2010 defines 'tolerance' as *'[...] difference between the upper limit of size and the lower limit of size [and is] an absolute quantity without sign'*. At the same place, 'tolerance limits' are defined as *'[...] specified values of the characteristic giving upper and/or lower bounds of the permissible value'*. The corresponding 'tolerance interval' is not necessarily symmetric. Variation is the total of all possible deviations of a measurable dimension. Practically speaking, the variation should be inside the tolerance interval in order to get a capable manufacturing process.

Tolerancing methods consist of a wide field of support methods to ensure quality, functionality, manufacturability and thus cost minimization by paying respect to geometrical product tolerances and their interrelation with all relevant product life cycle aspects. This work can only provide a first insight into these methods[25]. To start, within this section, tolerancing-related terminology and standards are presented for *tolerance specification* in technical drawings.

The following sections 2.2 and 2.3 demonstrate basic *tolerance calculation* methods for both 1D and 3D *tolerance analysis* using a simple case example. During the last three decades, tolerancing and all related aspects came more and more into focus within the international field of engineering research. Section 2.4 provides an overview about this and outlines interrelations between multiple methods to support product development processes. Another focus lies on *tolerance synthesis* methods which try to anticipate manufacturing tolerances of single parts or components in order to enable assembly tolerance calculation subsequently, as section 2.5 shows.

Aside methods for tolerance specification or actual tolerance calculation respectively tolerance synthesis methods, *tolerance management* stands for the approach to deploy engineering processes embedding tolerancing methods in the industrial product development process. Section 2.6 uses aircraft cabin tolerancing efforts to sketch a working process example for tolerance management activities, setting the focus on the core aspects of the present dissertation.

2.1.1 Tolerance Specification

Every dimension with functional influence should be specified with both its nominal value and the corresponding tolerance value in a technical drawing [82]. According to ISO 286-1:2010 for linear and angular measurements corresponding to features of size, so-called size tolerance specification is done with an individual tolerance indication directly at the location of the nominal dimension or with general tolerances (e.g. ISO 2768-1:1989).

However, size tolerance specifications can be ambiguous and do not specify shape, orientation or location tolerances for the feature. In the example given with figure 2.1 only the hole and pin diameters of a block and plate are specified with size tolerances[26].

[25] KLEIN'S books [81, 82] can be recommended for a detailed overview over tolerancing methods and the associated terminology including standardization.

[26] The example of the blocks and the plate introduced in fig. 2.1 on the next page will be used as example throughout this whole work, in particular for the tolerance analyses in sections 2.2 and 2.3.

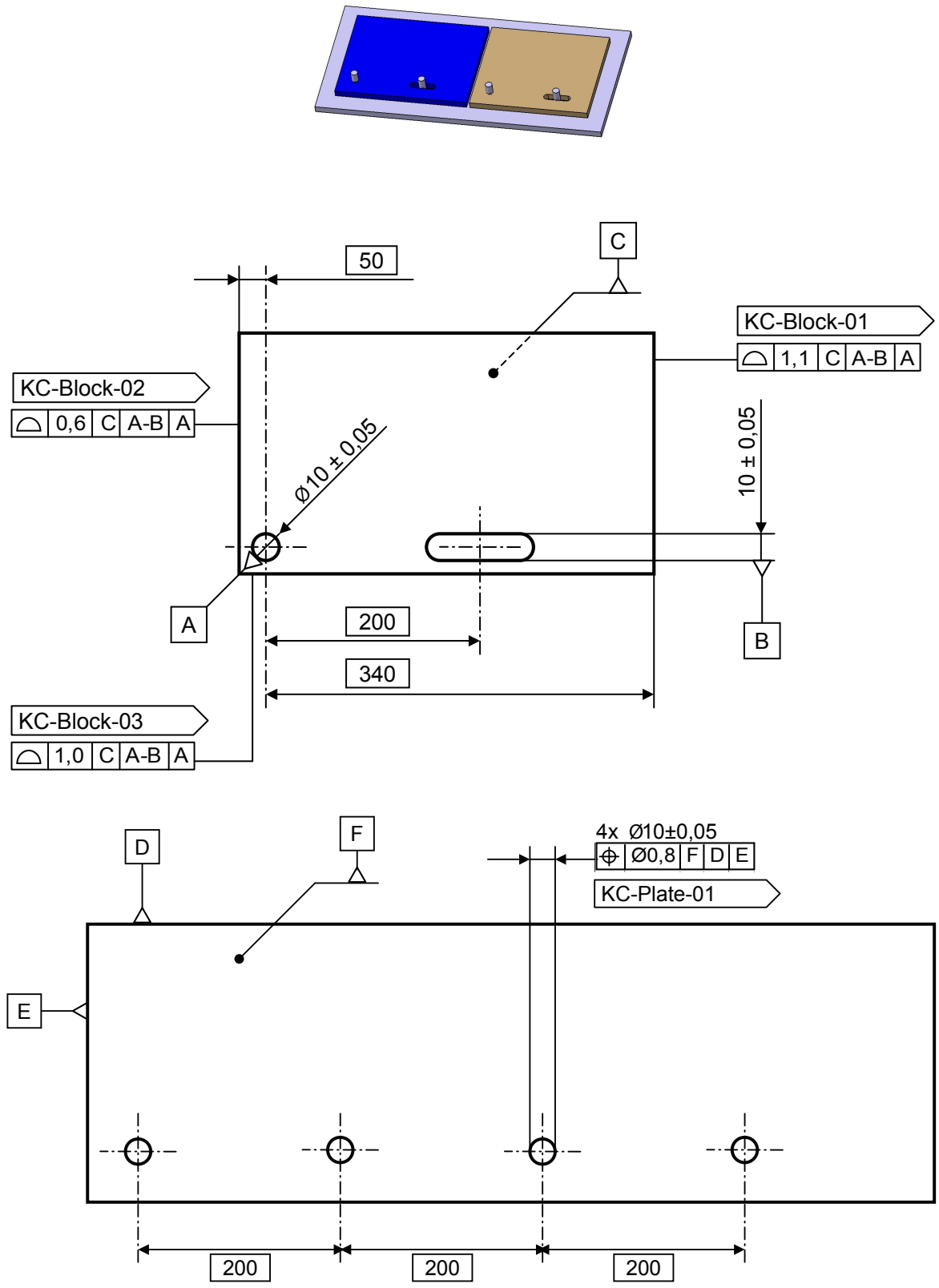

Figure 2.1: *Tolerance specification drawing for an assembly consisting of two blocks (upper drawing) and a plate (lower drawing) using size and GPS tolerances (see appendix A for explanations). Most dimensioning specifications are omitted for simplification.*

Tolerance specification for shape, orientation, location and run-out is preferably done with so-called 'geometrical product specification' (GPS, ISO/TR 14638:1995) tolerancing. GPS is a symbolic language for engineering drawings established with ISO standards. Figure 2.2 lists some of the applicable standards for tolerancing in general and GPS tolerancing in particular. ISO 1101:2004 defines several GPS symbols[27] for geometric tolerance specification. Using GPS tolerancing symbols, it is possible to assign a 'tolerance zone' to a geometrical feature. According to ISO 1101:2004, a 'tolerance zone' is a *'[...] space limited by one or several geometrically perfect lines or surfaces, and characterized by a linear dimension, called a tolerance [...]'*. The tolerance zone can be the space within a circle, the space between two concentric circles, the space between two equidistant lines or two parallel straight lines, the space within a cylinder, the space between two coaxial cylinders, the space between two equidistant surfaces or two parallel planes or the space within a sphere (ISO 1101:2004). Figure 2.1 shows several GPS tolerancing symbols to specify geometrical features of the two parts.

Standard	Title
ISO 286-1:2010	ISO system of limits and fits – Part 1: Bases of tolerances, deviations and fits
ISO 1101:2004	Geometrical Product Specifications (GPS) – Geometrical tolerancing – Tolerances of form, orientation, location and run-out
ISO 1660:1987	Technical Drawings – Dimensioning and tolerancing of profiles
ISO 2692:2006	Geometrical product specifications (GPS) – Geometrical tolerancing – Maximum material requirement (MMR), least material requirement (LMR) and reciprocity requirement (RPR)
ISO 2768-1:1989	General Tolerances – Part 1: Tolerances for linear and angular dimensions without individual tolerance indications
ISO 5458:1998	Geometrical product specifications (GPS) – Geometrical tolerancing – Positional tolerancing
ISO 5459:2011	Technical drawings – Geometrical tolerancing – Datums and datum systems for geometrical tolerancing
ISO 10578:1992	Technical drawings – Tolerancing of orientation and location – Projected tolerance zone
ISO 10579:2010	Technical drawings – Dimensioning and tolerancing – Non-rigid parts
ISO/TR 14638:1995	Geometrical product specification (GPS) – Masterplan
ISO/TS 17450-1:2005	Geometrical product specifications (GPS) – General concepts – Part 1: Model for geometrical specification and verification
ISO/TS 17450-2:2002	Geometrical product specifications (GPS) – General concepts – Part 2: Basic tenets, specifications, operators and uncertainties

Figure 2.2: *Selection of ISO-standards concerning tolerance specification*

Research has been focused on the extension of ISO standards by SRINIVASAN [76, 153, 154], most recently on a unambiguous method to express GPS including key characteristics and tolerancing by DANTAN et al. [35, 36, 37, 38, 108]. LANTRIP [94] shows aerospace application examples of geometric dimensioning and tolerancing[28] (GD&T) for flexible contoured structures.

[27] For a detailed description of the different GPS tolerances and symbols, consider A. For more detailed explanations about form and orientation tolerances the books of KLEIN [82] and HENZOLD [69] are recommended.

[28] While the ISO standards talk of 'geometrical product specification' (GPS), ASME Y14.5-2009 uses the term 'geometrical dimensioning and tolerancing' (GD&T). There are differences between ISO and ASME concerning tolerance specification, but they are minor for many applications [29, 82].

2.1.2 Datums and Datum Systems

The fact that every geometrical feature with functional relevancy should be toleranced implies that a suitable set of 'datums' must be established in order to enable unambiguous tolerance specification and measurement using measurement machines or manual measurement means (ISO 5459:2011). A 'datum' is a theoretically exact geometric reference (such as axes, planes, straight lines, etc.) *'[...] selected to define the location or orientation, or both, of a tolerance zone [...]'* of a geometric feature (ISO 5459:2011). 'Datum systems' are groups of two or more separate datums used as a combined reference for a toleranced feature. An ideal datum system of a component consists of three datums, which are called 'primary', 'secondary' and 'tertiary datum' [82], as figure 2.3 shows.

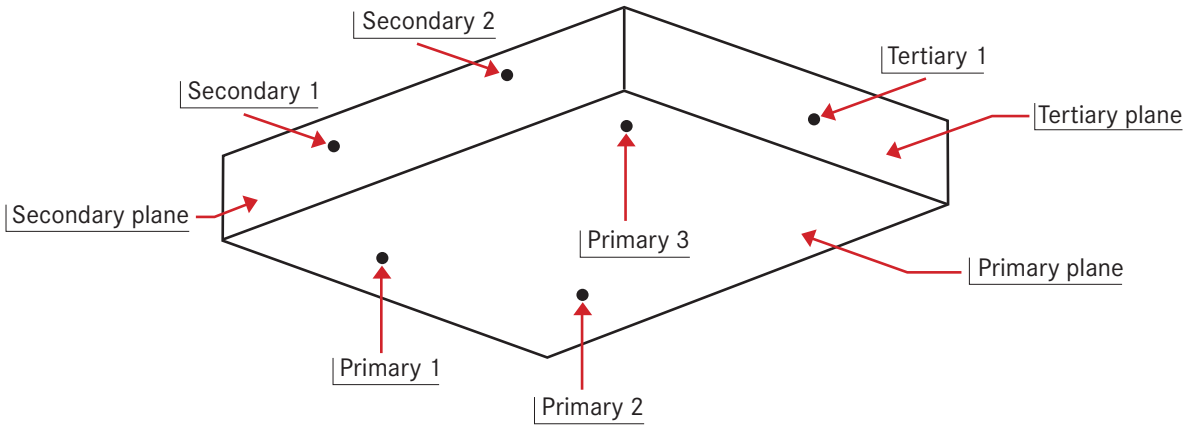

Figure 2.3: *A '3-2-1 datum system' according to* KLEIN *[82]*

Datums need to be defined by one or more 'datum features' (ISO 5459:2011). Six datum features are needed to define an isostatic datum system with a '3-2-1 datum system' [82], as can be seen in figure 2.3. The principle of 'plane-line-point' gives the name to the 3-2-1 datum system. The *primary plane* is constituted by exactly *three* datum features. More than three features would lead to a hyperstatic positioning, since only three points constitute a unambiguous definition of a plane. The primary plane blocks three degrees of freedom (DOF): one translatory DOF normal to the plane and two rotational DOF. The secondary datum is a *line* defined by *two* datum features, blocking two further DOF. Within measurement machines, these two points are used to span up the *secondary plane* perpendicular to the primary plane. More than two points for the definition of the secondary datum would lead to a hyperstatic description of the position within the measurement machine. The tertiary datum is *one* single feature (*point*) that blocks the remaining translatory degree of freedom. The point is used to span up the *tertiary plane* perpendicular to both other planes.

Datum features need to be a real feature of a part such as an edge, a surface or a hole to establish the location of a datum[29]. 'Theoretical' features like the center of a cutout or a planar shape have to be avoided as datum, because they are difficult to 'reach' in a unambiguous way, may require additional tooling with additional tolerances or may be subject to additional form tolerances as well. It is therefore

[29] Very often, those geometry feature are used as datum features which are also used for the attachment of the component within the assembly – e.g., brackets, holes, slots, contact planes etc. – each being a location where kinematic DOF are blocked between the component and the assembly (see section 2.3 for more detailed explanations). Heuristics and experience tell that this procedure minimizes assembly tolerances for most type-I assemblies [104, 170]. The cabin lining panels that are discussed later in this thesis and which are mostly installed using type-I installation processes use this principle, as it will be shown in chapters 4 and 5.

important to realize that purely virtual or digital elements like coordinate systems[30] in CAD files cannot be used as datum for tolerancing.

Every single part or sub-assembly needs its own assembly datum system. During assembly, the change of the datum systems might become necessary, which is called 'datum shift' according to FARMER [51]. Every change of a local datum system influences location, orientation and size of tolerance zones.

Within the example of figure 2.1, some of the mentioned GPS symbols including the datum specifications can be seen. For instance, the position symbol is used to locate the four pins relative to the primary datum F, the secondary datum D and the tertiary datum E of the plate within a circular tolerance zone. The surface symbol is applied to define the location and the orientation of the side surfaces of the block relative to the primary datum C, the secondary datum A-B (constituted by two datum features) and the tertiary datum A within a tolerance zone constituted by two equidistant surfaces. In order to simplify the example, most dimensioning specifications are left out.

2.1.3 Key Characteristic

EN 9100:2010 proposes the concept of 'key characteristics'. Herein, a key characteristic (KC) is defined as an *'[...] attribute or feature whose variation has a significant effect on product form, fit, function, performance, service life or producibility, that requires specific actions for the purpose of controlling variation [...]'*. Among others, WHITNEY [169], LEE [98] and MARGUET [106] explain experiences with the usage of KCs for tolerancing[31].

KCs usually need to be measured throughout the whole product life cycle[32]. KCs and the associated controlling processes are described in EN 9100:2010 and EN9103:2005. The concept is used to name and identify particular geometrical features serving a functional purpose. Usually, KCs are specified using GPS symbols and get an additional KC flag with a name tag for a unambiguous identification. The example given with figure 2.1 also shows some KC flags.

2.2 1D Tolerance Analysis

Basically, tolerance analysis either follows a worst-case approach or statistical calculation methods for so-called 1D tolerance stack analysis. Aside KLEIN [81, 82], the authors CHASE [22], MEADOWS [112] and SCHOLZ [143, 144] all provide contributions to gain an overview.

2.2.1 Worst-Case Analysis

The so-called 'worst-case analysis' [22, 82] or also '1D stack analysis' [112] is an arithmetical consideration of an assembly's tolerances. All tolerance contributors t_i are summed up to the total assembly tolerance T_{wc}.:

$$T_{wc} = t_1 + t_2 + \cdots + t_n \tag{I}$$

T_{wc} is the tolerance according to ISO286-1:2010 for the variation of the assembly feature. As long as the contributing tolerances t_i are centered around the nominal value N, T_{wc} is centered as well. This means that the final gap size G can vary within the measurable interval $[N - \frac{T}{2}; N + \frac{T}{2}]$ with the tolerance limits

[30] *CAD coordinate systems* and *tolerancing datum systems* are two different and independent reference systems for different purposes. A *CAD coordinate system* is a *virtual* reference with a virtual origin point and virtual axes. It is used for CAD models as an auxiliary concept. A *tolerancing datum system* is a *real* reference constituted by *real* geometry features and is used to define an explicit and unambiguous positioning of a part for tolerance specification and measurement [82].

[31] See section 2.6 for some aircraft cabin-specific definitions.

[32] In order to reduce measurement efforts, the concept of 'critical items' according to EN 9100:2000 can be used instead. Critical items only *'[...] require specific actions to ensure they are adequately managed [...]'*, which can be interpreted as meaning that no general measurement is required.

$\pm\frac{T}{2}$. In case of non-centered tolerances, the upper and the lower worst-case limits need to be calculated separately, as explained in detail in the above mentioned literature. For centered tolerances, mostly the following variant of equation I is used for practice:

$$\pm\frac{T_{wc}}{2} = \pm\left(\frac{t_1}{2} + \frac{t_2}{2} + \cdots + \frac{t_n}{2}\right) \tag{II}$$

As long as all contributors remain within their respective tolerance, the worst-case calculation ensures that all assembly combinations will have a deviation which is as good as or better than the calculated assembly tolerance T_{wc}. The values of the tolerance contributors t_i either are based on experience including measurement data, heuristics or on general tolerances (e.g., ISO 2768-1:1989) or have to be derived from dedicated tolerance synthesis methods[33].

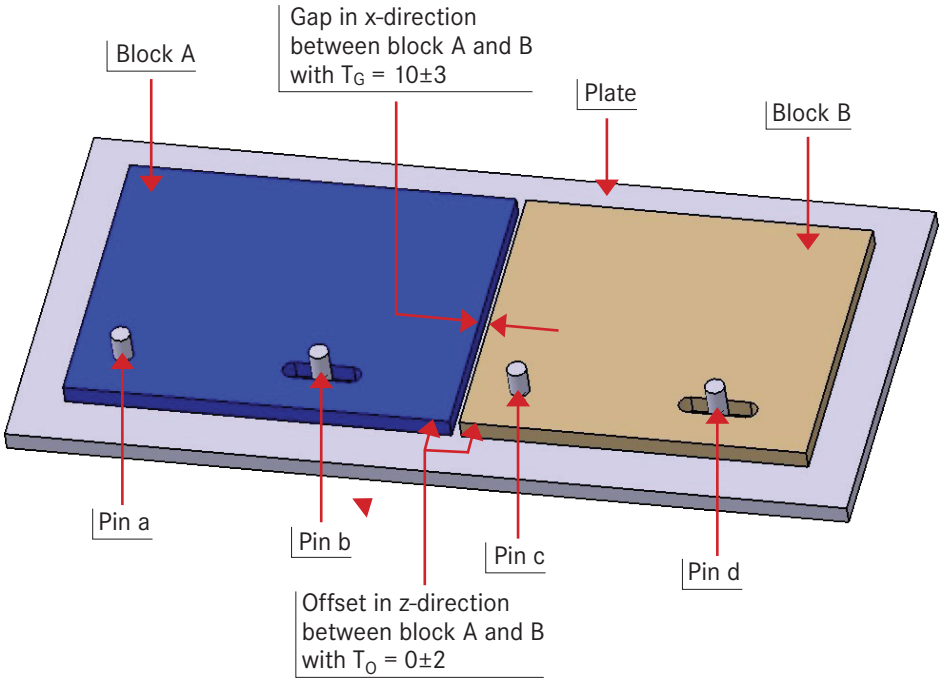

Figure 2.4: *Assembly tolerance analyses of interest for the example introduced with fig. 2.1*

Figure 2.4 shows the well-known example with the two blocks A and B fixed to the base plate P with two interfaces, namely a '4-way locator' and a '2-way locator'[34] each. The corresponding tolerance specifications are given in figure 2.1. Between the two blocks there is a gap with the nominal size $N = 10mm$. Due to appearance quality reasons, the tolerance limits $\pm\frac{T_G}{2} = \pm 2.0mm$ corresponding to a centered tolerance interval of $T_G = 4mm$ have been specified[35].

The 1D tolerance stack belonging to the Gap between block A and B starts at the lower right edge of block A at the left side of the Gap. From here, there is a tolerance of $t_1 = 1.1$ (specified as particular

[33] See section 2.5 for information about tolerance synthesis.

[34] These terms are used to describe the functional behavior of attachment interfaces. A '4-way locator' means that at this location relative movement between two parts is locked 'forwards and backwards' as well as 'left and right'. A simple example for such an interface is a pin-hole combination (e.g., fig. 2.1), where the tolerance zones for both the pin and the hole are usually circular. A '2-way locator' locks the movement in two directions (e.g., 'left and right'). An example is a pin-slot combination (e.g., fig. 2.1).

[35] In the following the notation of the dimension 'millimeter' (mm) of the tolerances will be left for simplification reasons.

Key Characteristic named 'KC-Block-01' in fig. 2.1) to the far left fixation pin a. Between pin a and the hole of plate A, a game of $t_2 = 0.2$ is considered[36]. From pin a to datum D of the plate and from there to pin c, two times a tolerance of $t_3 = 0.8$ has been specified ('KC-Plate-01' valid for all pins). The game tolerance of $t_4 = 0.2$ is considered again at pin c, from where a tolerance of $t_5 = 0.6$ applies from the hole to the left edge of plate B ('KC-Block-02'). Now, having reached the right side of the gap G, the tolerance stack is complete. Summing up the tolerances according to equation II leads to

$$\pm \frac{T_{G,wc}}{2} = \pm \left(\frac{t_1}{2} + \frac{t_2}{2} + 2 \cdot \frac{t_3}{2} + \frac{t_4}{2} + \frac{t_5}{2} \right) = \pm 1.85. \tag{III}$$

This resulting deviation is below the tolerance limit of ± 2.0, which means that the requirement will be fulfilled any time. As long as the contributing components are manufactured within their specified tolerances t_i, no assembly combination can fail.

2.2.2 Statistical Analysis

The probability that worst-case situations occur usually is very small. To reduce manufacturing costs, it may be of interest to increase the required manufacturing tolerance limits $\pm \frac{t_i}{2}$ if it can be ensured that the resulting assembly tolerance T will be within the required limits anyway – at least with an adequately high probability.

For 'statistical tolerancing' [81, 82, 143, 144], which is the most prevalent method to perform manual statistical tolerance analysis, three prerequisites are important[37]:

1. The tolerances' variation has to follow a Gaussian distribution.
2. The distribution curve has to be centered.
3. The variation of the involved tolerances has to be statistically independent.

Using these assumptions, 3D assembly tolerances can be calculated with the so-called root square sum (RSS) formula [143]:

$$T_{RSS} = \sqrt{t_1^2 + t_2^2 + \cdots + t_n^2} \tag{IV}$$

Furthermore, T_{RSS} follows a Gaussian distribution of the same σ-accuracy as the input tolerances t_i. For the most part, a $\pm 3\sigma$-accuracy corresponding to 99.73% of good parts is assumed for conceptual work. Since centered contributors are prerequisites, the symmetric tolerance limits $\pm \frac{T}{2}$ respectively $\pm \frac{t_i}{2}$ can be used alternatively to the tolerance intervals T and t_i, leading to

$$\pm \frac{T_{RSS}}{2} = \pm \sqrt{\left(\frac{t_1}{2}\right)^2 + \left(\frac{t_2}{2}\right)^2 + \cdots + \left(\frac{t_n}{2}\right)^2}. \tag{V}$$

If statistical tolerancing is used but the mentioned prerequisites are violated, there is a high risk that the calculation results are too optimistic. However, during the engineering design phase it is difficult to predict robust variation and process capabilities while ensuring Gaussian distributions and statistical independence. Therefore it proves valuable to use safety factors for the RSS for conceptual tolerancing to decrease the probability of assembly deviations outside the calculated tolerance instead of going back

[36] The range of $t_2 = 0.2$ corresponding to $\pm \frac{t_2}{2} = \pm 0.1$ comes from the sum of the diameter tolerance (feature tolerance) of hole a of the block with ± 0.05 and the corresponding pin diameter tolerance of ± 0.05, see fig. 2.1.

[37] In practice, manual statistical tolerance analysis is hardly done without 'statistical tolerancing' as method (except for some empirical extensions or corrections of the RSS formula). Of course, it is possible to perform statistical tolerance analysis beyond these assumptions, as it is mentioned later at the beginning of section 2.3 and in subsection 2.3.3 on simulation-based tolerance analysis. However, the fixed term 'statistical tolerancing' [81, 82, 143, 144] always refers to these assumptions.

to pure worst-case analysis. The widely-used safety factor of 1.5 is often called the BENDER-factor [143, 144]. Originally, it comes from the intention to provide a $\pm 3\sigma$-accuracy for the target T assuming the input tolerances t_i have a $\pm 2\sigma$-accuracy only [143, 144].

Within industrial practice, this factor or similar factors of comparable size are still widely used. Since the assumption of a $\pm 3\sigma$-variation is convenient practice nowadays, the safety-factor or BENDER-factor is often re-interpreted as a compensation factor for unknown mean shifts of the contribution tolerances t_i. The intention to foresee and to compensate for mean shifts depends on the toleranced object. Gaps for aesthetic quality may still 'work' if their mean is not exactly their nominal value as long as the parallelism of the gaps is not affected. In contrast, for installation tolerances like the coincidence of a pin and a slot this can lead to intense installation and design problems. For calculation formulas considering distribution curves other than the Gaussian distribution and for calculations considering means-shifted tolerances in a formal way including the meaning and the usage of the process capability indices C_p and C_{pk}, one should refer to the corresponding literature like that cited at the beginning of this section. See also section 2.5 for links to quality assessment methods.

Going back to the example given by figure 2.4, it is assumed that it has been decided to reduce the tolerance limits to $\pm\frac{T_G}{2} = \pm 1.0$ in order to improve the appearance quality. Additionally, it is supposed that a synthesis approach using the worst-case algorithm has shown that smaller manufacturing tolerances are either too expensive or are not feasible with capable manufacturing processes, but that current tolerances can be considered as capable for centered independent $\pm 3\sigma$-processes. Now equation V can be used to calculate the resulting value:

$$\pm\frac{T_{G,RSS}}{2} = \sqrt{\left(\frac{t_1}{2}\right)^2 + \left(\frac{t_2}{2}\right)^2 + 2\cdot\left(\frac{t_3}{2}\right)^2 + \left(\frac{t_4}{2}\right)^2 + \left(\frac{t_5}{2}\right)^2} \approx \pm 0.86 < \pm 1.00. \qquad \text{(VI)}$$

Relation VI can be interpreted such, that approximately 99.73% of all assemblies are within the calculated tolerance – as long as the components stick to the named prerequisites for statistical calculations. The percentage of good parts is even higher, as the calculated result is significantly below the specified limits of ± 1.0. If the assumption of centered distribution curves proves to be too optimistic and the recommended safety factor of 1.5 is used consequently to cover for mean shift effects, the result turns out to be $\pm\frac{T_{G,1.5xRSS}}{2} = \pm 1.28$. This result would no longer be adequate to ensure a capable process for the updated requirement.

It has to be mentioned again that the safety factor of 1.5 here only covers a certain magnitude of the mean shift, which hardly can be anticipated before manufacturing starts. The real uncertainty of the calculated result ('the error of the expected deviation') may increase with the number of applied hypotheses. Frankly, 'tolerance analyses' [82] here turn more into 'tolerance estimations' or into 'concept checks' rather than providing a 'definite' or 'scientifically sound' tolerance result. It is therefore recommended to start iterative processes at this stage with calculation, measurement and design requirement adaptations before freezing tolerance targets too early based on too many hypotheses.

2.3 3D Tolerance Analysis

3D tolerance analysis is much more labor-intensive than 1D stack analysis. For this reason, there is plenty of work on commercial and academic computer-aided tolerancing (CAT) tools for 3D tolerance analysis [105, 126, 137]. WANG [165] provides a more recent overview about ongoing research, focusing on the different modeling concepts. In the industrial context, usually two methods are in use, which are based either on a kinematic model using small displacements [17, 26] or direct linearization[38] [20, 21, 23] or on Monte Carlo simulations [54].

Taking a look at the research sector, alternative tolerance analysis methods and models beyond statistical tolerancing are under investigation [19, 165]. SRINIVASAN et al. describe the method and tool development work using fractal-based parameters for tolerance calculation [152]. DAVIDSON et al. build up a mathematical model to formalize tolerance specifications using 'tolerance-maps' [6, 39]. BALLU proposes a method to analyze hyperstatic mechanisms using first and second order reliability method algorithms [13]. KHODAYGAN discusses tolerance analysis of assemblies based on 'fuzzy logic' [80].

Generally speaking, 1D tolerance analysis is still state of the art in the industrial context for many use cases, often by using calculation spreadsheets. If 3D tolerances need to be considered, this still increases calculation complexity and time compared to sole 1D analysis. Consequently, there are tendencies towards more CAE interconnection and modeling automatization within commercial CAT software. The user interfaces and the functionality of the tools are continuously extended, for instance by links to deformation simulation or to cost analysis. Similarly, in this thesis a method for integrated and automated 3D tolerance analysis is developed.

In order to provide a more detailed insight into the application of CAT software, subsequently two commercial software tools will be explained a bit more closely – the first one, MECAmaster, based on the small displacement theory [26, 113] in subsections 2.3.1 and 2.3.2, the second one, 3DCS, based on Monte Carlo simulation algorithms [42, 52] in subsection 2.3.3.

2.3.1 Small Displacement Theory

Considering that tolerances are usually very small compared to the nominal dimensions, linearized mathematical methods can be applied [17, 20, 21, 23, 26]. The corresponding modeling assumptions are:

- consideration of 3D tolerance problems as directed or vectorial problems (GPS tolerances),
- representation of the attachment principle and assembly sequence by a non-ambiguous *kinematic linkage system*[39],
- linearized calculation of the *influence coefficients* $c_{j,i}$, neglecting the non-linear influence of the coefficients on each other,
- linear superposition of weighted component tolerances $c_{j,i} \cdot t_i$ in order to calculate the target tolerances T_j using linearized displacements,
- all components considered rigid,
- all components mounted with isostatic or simple hyperstatic[40] attachment concepts.

[38] See also the web page of ADCATS (Association for the Development of Computer-Aided Tolerancing Systems), http://adcats.et.byu.edu, accessed Jan 2012.

[39] *Kinematic linkages* are those locations of an assembly where the kinematic DOF are blocked between the components. The *kinematic linkage system* of a component is consequently the set of linkages which blocks all six DOF of this component relative to the remaining assembly.

[40] 'Simple hyperstatic' can mean, e.g., that the primary attachment plane is constituted by more than three points, for which simplifying calculation algorithms can be used [113].

Without the last two assumptions, the analysis could lead to non-linearizable higher-order deformation problems. All together, the assumptions allow to make 3D tolerance calculations with simple computational efforts. Firstly, this means that the orientation of the tolerance zone of a feature has to be considered, which is defined by its orientation relative to the corresponding datum system of the component. As consequence, the number of contributing tolerances to a 3D tolerance stack may increase compared to a 1D calculation[41]. Secondly, the contribution of a feature tolerance to a tolerance stack may be $\neq 1$ due to geometric lever arm effects. This is implied by the position of a feature relative to the component's kinematic linkage system, as shown later in the example given by figure 2.5.

Both phenomena together lead to so-called *influence coefficients* $c_{j,i}$, with which the tolerance contributors t_i have to be multiplied in the linearized 3D tolerance stack for the assembly tolerance T_j. The index i stands for the contribution tolerance and j for the tolerance target. The theory states, that any interface between the involved components – specifically called 'kinematic linkages' or simply 'linkage' [27, 28, 113] – with its corresponding feature tolerance(s) can contribute to any target tolerance, depending on the influence coefficient $c_{j,i}$.

This leads to the matrix equation [27]

$$T = C \times t \tag{VII}$$

or

$$\begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{pmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & & \vdots \\ \vdots & & \ddots & \vdots \\ c_{m,1} & \cdots & \cdots & c_{m,n} \end{bmatrix} \times \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}. \tag{VIII}$$

For a worst-case problem with one target tolerance ($m = 1$), this results in one single equation for worst-case analysis given with

$$\pm \frac{T_{3D,wc}}{2} = \pm \left(c_{1,1} \cdot \frac{t_1}{2} + c_{1,2} \cdot \frac{t_2}{2} + \cdots + c_{1,n} \cdot \frac{t_n}{2} \right), \tag{IX}$$

as well as in

$$\pm \frac{T_{3D,RSS}}{2} = \pm \sqrt{\left(c_{1,1} \cdot \frac{t_1}{2} \right)^2 + \left(c_{1,2} \cdot \frac{t_2}{2} \right)^2 + \cdots + \left(c_{1,n} \cdot \frac{t_n}{2} \right)^2} \tag{X}$$

for statistical analysis assuming centered distributions.

In the 1D case, the factors c_i are either equal to 1 or to 0, leading back to equation I or IV. In general, theoretically all $c_{j,i}$ could be implicitly dependent on the other influence coefficients, as a tolerance-caused shift or rotation of a part can change the geometric relations. With the linearization approach, this phenomenon is neglected and all tolerances are superposed in a linear way. This also means that both input and output tolerances must be considered as independent. Each tolerance t_i can additionally consist of several contributors: for instance one for the positional tolerance of the first component at the kinematic linkage, one for positional tolerance of the second component and one or more for interface-related tolerances, such as a game between a pin and a hole or interface flexibility due to soft materials.

[41] FARMER calls this phenomenon 'datum shift' [51].

Going back to the previous example with the two blocks and the plate, now the offset T_O in z-direction between the two blocks at their lower inner edge points comes into focus (see fig. 2.4). The requirement is ± 3. Within the assembly twelve degrees of freedom (DOF) between the three components are blocked – six DOF per block relative to the plate. Due to the planar type of the assembly, for each block the translational degree of freedom perpendicular to the contact plane and the two related rotational degrees of freedom hardly have any influence in the offset T_O. The corresponding influence coefficients are almost 0 and are neglected in the following in order to simplify the calculation.

Consequently, there are six remaining kinematic linkages with influence on the offset T_O and therefore six different linearized influence coefficients need to be calculated plus one for the feature tolerances at the measurement point itself. As only one target tolerance is investigated, m becomes $m = 1$ and the influence coefficients $c_{1,i}$ or simply c_i are referred to as

- c_{ax} for the blocked translational DOF in x-direction at pin a,
- c_{az} for the blocked translational DOF in z-direction at pin a,
- c_{bz} for the blocked translational DOF in z-direction at pin b,
- c_{cx} for the blocked translational DOF in x-direction at pin c,
- c_{cz} for the blocked translational DOF in z-direction at pin c,
- c_{dz} for the blocked translational DOF in z-direction at pin d and
- c_m for the influence of the feature tolerances at the measurement point[42].

The corresponding worst-case calculation is

$$\begin{aligned} &\pm \frac{T_O, wc}{2} = \\ &\pm \left(c_{ax} \cdot \frac{t_{ax}}{2} + c_{az} \cdot \frac{t_{az}}{2} + c_{bz} \cdot \frac{t_{bz}}{2} + c_{cx} \cdot \frac{t_{cx}}{2} + c_{cz} \cdot \frac{t_{cz}}{2} + c_{dz} \cdot \frac{t_{dz}}{2} + c_m \cdot \frac{t_m}{2} \right). \end{aligned} \tag{XI}$$

Again, the contribution tolerances t_i can be read out of the drawing within figure 2.4. At pin a, block A has its datum A in x-direction at this location, so the tolerance is 0. The interface provides a game of ± 0.1 or 0.2 in range[43]. The pin at the plate has a positional tolerance of 0.8 respectively of ± 0.4 in x-direction[44], leading to $\frac{t_{ax}}{2} = 0 + 0.1 + 0.4 = 0.5$. Datum A also acts in z-direction, leading to a contribution of 0, too. Again the interface and the pin contribute with ± 0.1 and ± 0.4 to $\frac{t_{az}}{2} = 0 + 0.1 + 0.4 = 0.5$. The same is valid for $\frac{t_{bz}}{2} = \frac{t_{cx}}{2} = \frac{t_{cz}}{2} = \frac{t_{dz}}{2} = 0 + 0.1 + 0.4 = 0.5$. The feature tolerance at the measurement point is two times 1.0, once for each plate[45]. Hence, $\frac{t_m}{2}$ becomes $\frac{t_m}{2} = 0.5 + 0.5 = 1.0$.

[42] In general, the influence coefficient c of a measurement point can be $\neq 1$ since the direction of the feature's tolerance zone and the direction of the measurement are not always identical.

[43] Compare with footnote 36, page 25.

[44] 'KC-Plate-01', see fig. 2.1, page 20.

[45] 'KC-Block-03', see fig. 2.1, page 20

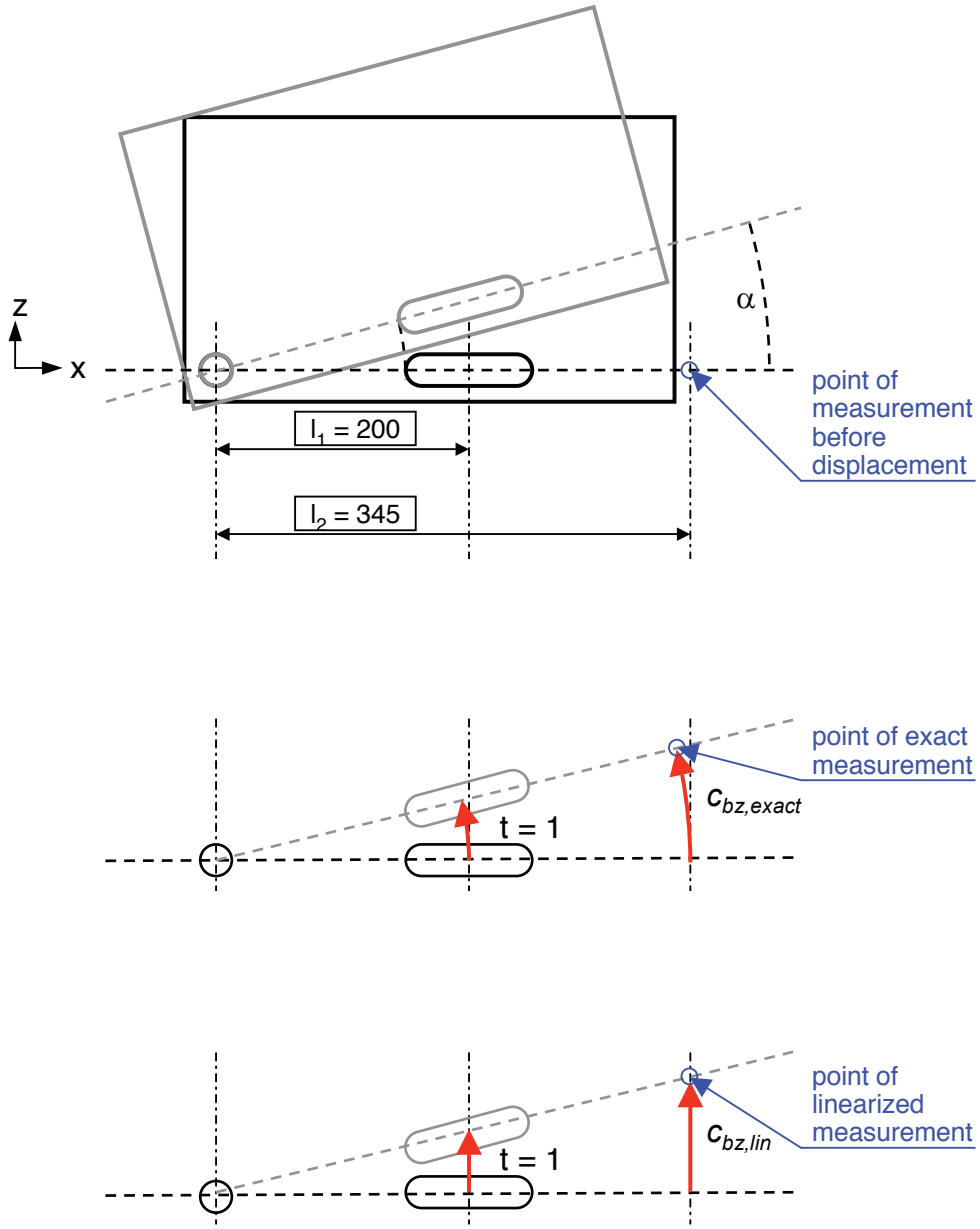

Figure 2.5: *The influence coefficient c_{bz} corresponding to the tolerance t_{bz} in z-direction at pin b due to the rotation by the angle α around pin a. It can be seen, that the linearization of this specific problem does not take into consideration the small displacement [27] in x-direction at the measurement point due to the displacement in z-direction at the 2-way locator.*

When block A rotates around pin a by the angle α due to an unit displacement $t_{bz} = 1$ at pin b, the lower right edge point moves upwards with a magnitude of c_{bz}. If the measurement is taken at the middle of the gap, l_2 becomes $l_2 = 345$. The exact calculation with equation XIV using equations XII and XIII leads to $c_{bz,exact} = 1.72498$ (see fig. 2.5).

$$\sin\alpha = c_{bz}/l_2 \tag{XII}$$

$$\tan\alpha = \frac{1}{l_1} \tag{XIII}$$

$$c_{bz,exact} = l_2 \cdot \sin\left(\arctan\frac{1}{l_1}\right) \tag{XIV}$$

For the linearized assumptions of small displacements [27], $\arctan\alpha \approx \alpha$ and $\sin\alpha \approx \alpha$ are used [18], providing a coefficient $c_{bz,lin} = \frac{l_2}{l_1} = \frac{345}{200} = 1.725$ (see eq. XV). The linearized result is only 0.00125% larger than the exact value – an order which is negligible for most applications, especially when additional geometry simplification assumptions are made.

$$c_{bz,lin} \approx l_2 \cdot \sin\left(\frac{1}{l_1}\right) \approx l_2 \cdot \left(\frac{1}{l_1}\right) = \frac{l_2}{l_1} \tag{XV}$$

If a displacement is applied at pin a in z-direction, block A rotates around the pin b, and the lower right edge point moves downwards in z-direction. The linearized calculation provides $c_{az} = \frac{145}{200} = 0.725$. If there is a displacement at pin c in z-direction, the close-by lower left edge faces a displacement of $c_{cz} = \frac{55+200}{200} = 1.275$. Due to the rotation around the pin c, a displacement at pin d has a small effect on the edge point, namely $c_{dz} = \frac{55}{200} = 0.275$. In a linearized calculation, neither at pin a nor at pin c does a displacement in x-direction have any effect on the offset in z-direction, which means that c_{ax} and c_{cx} become $c_{ax} = c_{cx} = 0$. The influence of the feature tolerances at the measurement is $c_m = 1$, since the direction of the tolerance zone at this edge is in the (linearized) direction of measurement of the target tolerance T_O.

When the results for the influence coefficients c_i and for the contribution tolerances t_i are inserted into equation XI, the worst-case result for T_O becomes $\pm\frac{T_{O,wc}}{2} = \pm 3.00$. The RSS result is $\pm\frac{T_{O,RSS}}{2} = \pm 1.177$, the safety-factored result is $\pm\frac{T_{O,1.5xRSS}}{2} = \pm 1.765$. Depending on design- and manufacturing-related boundary conditions, the results now need to be interpreted and discussed in the industrial context.

This example demonstrates the efforts to conduct a simple 2D tolerance analysis. Even with the linearization simplifications, the calculation of the influence coefficients can become sophisticated for complex assemblies. That is why manual 3D tolerance calculation is very time intensive and is usually done with the help of CAT software.

2.3.2 Implementation in MECAmaster

MECAmaster [113] is a commercial CAT software using the small displacement theory as modeling approach [27, 28]. The MECAmaster modeling concept considers the assembly sequence as the order in which geometrical constraints are established during the different assembly steps, defining in particular when, where and how the DOF of the assembled parts are fixed. The modeled linkage systems consequently depend on the physical attachment principles and on the assembly sequence of the involved components. However, unlike in simulation methods for tolerance analysis, the sequence is not modeled directly or explicitly, but is reflected implicitly in the modeled *kinematic linkage system*.

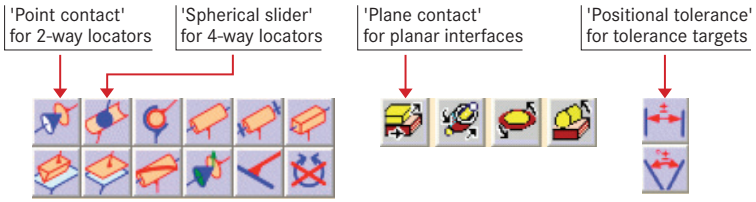

Figure 2.6: *MECAmaster objects for to model kinematic linkages and tolerance targets*

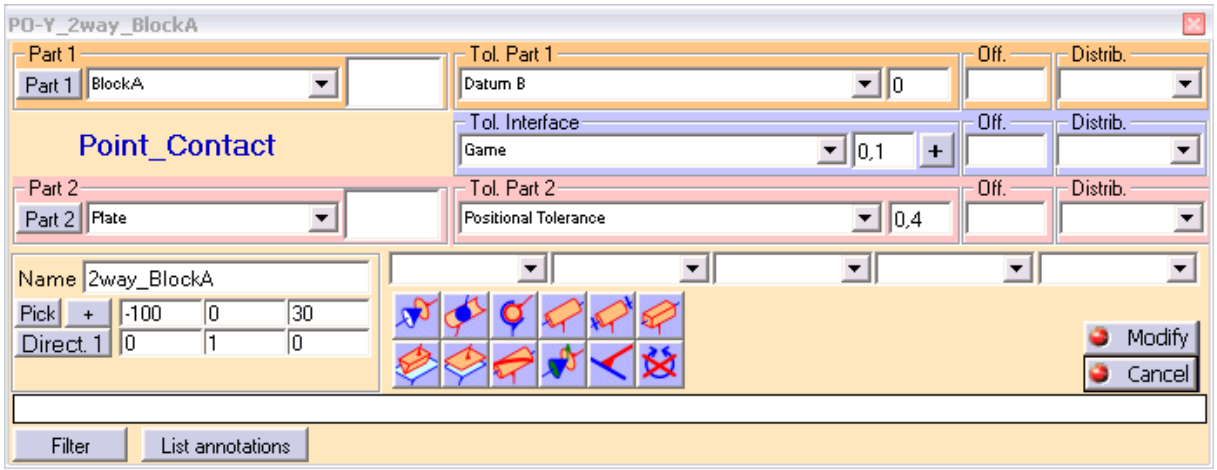

Figure 2.7: *MECAmaster GUI to input a 'point contact' element and the corresponding tolerances.*

The MECAmaster modeling or data elements are basically 'linkages' and the 'positional tolerances' (fig. 2.6). 'Positional tolerances' can be considered as 'linkages' without kinematic constraint, but with a spatial distance requirement. Amongst others[46] [113], the most important elements for this work are:

- *Positional tolerance:* tolerance target for a gap or split line between two components,
- *Point contact:* linkage blocking one translational DOF, e.g., used for 2-way locators,
- *Spherical slider:* linkage blocking two coupled translational DOF, e.g., used for 4-way locators,
- *Plane contact:* group of coupled linkages blocking one translational and two rotational DOF[47].

[46] Compare also with the overview about 'kinematic joints' from CHASE [23] given in subsection 1.1.3.

[47] 'Plane contacts' can consist of up to eight points. Mathematically, only three points are needed to span up a plane and any further point leads to an hyperstatic condition. Hence, an internal algorithm ensures, that for each tolerance calculation the 'most realistic' three points are used as linkages and the others are ignored in this case.

The working process using MECAmaster is as follows [113]:

1. GUI-based definition (e.g., fig. 2.7) of all required tolerance target values T_j (e.g., 'positional tolerances' as shown in fig. 2.6, visualized as turquoise arrows in fig. 2.8 or as blue lines with label 'PT' in fig. 2.9), which are internally compiled in the vector T.

2. GUI-based definition of the kinematic linkages L_i (e.g., 'point contacts' or 'spherical sliders' as shown in fig. 2.6, visualized as magenta symbols in fig. 2.8 or as magenta lines with labels 'PO' or 'SS' respectively in fig. 2.9) of the assembly by reducing the kinematic DOF to zero.

3. GUI-based definition of all contributing linkage feature tolerances[48] t_i, which are internally compiled in the vector t (e.g., fig. 2.7 shows the GUI to input a 'point contact' and the corresponding tolerances).

4. Automated linearized calculation
   (a) Automated calculation of the influence factor matrix C containing all linearized influence coefficients c_{ji} between every tolerance t_i and every target value T_j.
   (b) Automated calculation of all T_j by solving the matrix equation (eq. VIII). The weighted tolerance values $c_{ji} \cdot t_i$ are superposed with either the worst-case or the statistical method.
   (c) Output of the linearized 3D tolerance stacks, linearized influence coefficients (see fig. 2.10) and functional diagrams representing the modeled linkage system (e.g., see fig. 2.9).

Figure 2.8 shows the MECAmaster model of the well-known example with the blocks and the plate (see fig. 2.1 and 2.4). The two blocks are hidden for a simpler representation. Pins a and c are modeled using a 'spherical slider' (see magenta balls), while the interfaces at pins b and d are represented by 'point contacts'. In order to complete the full 3D model, 'plane contacts' are established in z-direction between the blocks and the plate. The two tolerances targets T_G and T_O are modeled accordingly as 'positional tolerances'. Figure 2.9 shows an alternative visualization of the same model. This abstract diagram is frequently used as basis for technical discussions about the chosen linkage systems and to check whether correct modeling input has been defined[49].

The resulting 3D stack output for the target tolerance shown in figure 2.10 validates the 1D calculation results from the previous subsection. For instance, the 3D stack for the target tolerance T_O leads to $\pm\frac{T_{O,wc}}{2} = \pm 3.0$, $\pm\frac{T_{O,RSS}}{2} = \pm 1.177$ or $\pm\frac{T_{O,1.5xRSS}}{2} = \pm 1.765$ respectively, which are exactly the same results compared to the manual calculation from above. Figures 2.10 also displays a colored representation of the weighted stack contributors $c_{ji} \cdot t_i$. This visualization can be a very helpful mean to detect and to understand the major contributing factors of 3D stacks[50].

The calculation of MECAmaster usually does not exceed several seconds. It makes sense to check the modeled linkage system and to compare it with modeling alternatives, if required. MECAmaster is used together with CATIA V5, but it is not an integrated workbench and has an independent GUI [113] (e.g., see fig. 2.7). Additionally, all input data is saved in data files separate from the CATIA tree.

[48] Composite tolerances can be used for repetitive tolerances like tooling tolerances. Group tolerances for patterns of small local tolerances are implemented.

[49] Fig. 6.4 on page 116 shows a much more complex model making clear why this abstraction is often used for discussions rather than using the complete – and even more complex – 3D model.

[50] From a user point of view, it is an advantage of kinematic tools like MECAmaster that the influence coefficients are calculated directly. Already without available tolerance information, the coefficients tell which feature or characteristic of which component influences the assembly tolerance very strongly ($c_{ji} >> 1$) and so needs special care. It also shows which coefficient has limited influence ($c_{ji} << 1$) and so even increased tolerance values would not impact the assembly. Colored visualizations like those given with fig. 2.10 support these analyses.

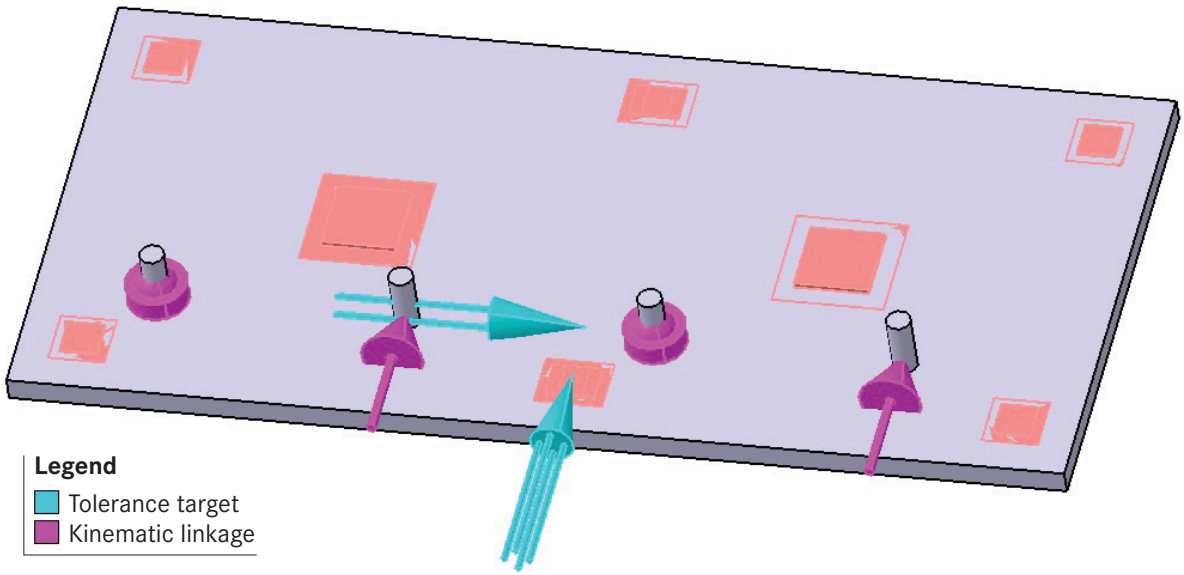

Figure 2.8: *MECAmaster model of the block and plate example. Magenta arrows represent 'point contacts', magenta spheres are 'spherical sliders', orange shapes are 'plane contacts' and turquoise arrows are 'positional tolerances'.*

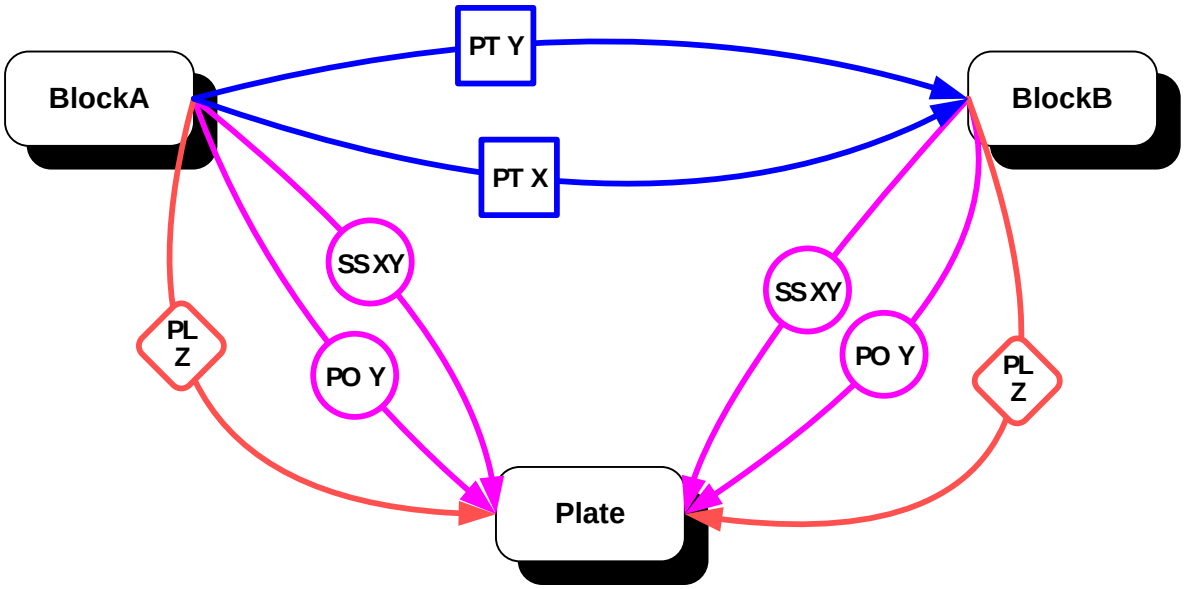

Figure 2.9: *Alternative visualization of the linkage system with a diagram function implemented in MECAmaster. The white blocks represent the three components as shown in fig. 2.8, the Magenta lines with 'PO' and 'SS' represent 'point contacts' or 'spherical sliders' respectively, orange lines with 'PL' are 'plane contacts' and blue lines with 'PT' are 'positional tolerances' in the corresponding functional directions. The MECAmaster linkage diagrams are used as basis for technical discussions about the chosen linkage systems and to check whether correct modeling input has been defined.*

```
--|------- Name -------|------ Parts  ------| Tolerance | Influence | Contribu
                        blocka                    .ooo \
 SS XY 4way_BlockA                          .1ooooo  x    1.ooo  =  .5ooooo
                        frame               .4ooooo /                27.0%
 ------------------------------------------------------------------------------
                        blockb                    .ooo \
 SS XY 4way_BlockB                          .1ooooo  x    1.ooo  =  .5ooooo
                        frame               .4ooooo /                27.0%
 ------------------------------------------------------------------------------
                        blocka              .55oooo \
 PT X  T_G                                          x    1.ooo  =  .85oooo
                        blockb              .3ooooo /                45.9%
 ------------------------------------------------------------------------------
 ------------------------------------------------------------------------------
 ARITHMETICAL calcul. (sorted contrib)      Value of the tolerance   =    1.85o

 /  For the linkages         \   / Value due to the parts defaults   =    1.65o
 \  exclusively in position  /   \ Value  due to the  interfaces     =  .2ooooo


 Value of the resulting tolerance :

 ATTENTION !   Statistical calculation for independant statist. data  (Gauss)
 ----------    Use the values with !!! just if you exactly know about them .
                              ---------------------------------------------
     Value of the tolerance:    |Probability  of having the  fixed tolerances |
       1 of the 9 values.       |    0.9544         0.9973         0.9999     |
 ------------------------------------------------------------------------------
 Wished Probability      0.9544 |    .855862   |    .570575!!!|    .427931!!! |
 to get the              0.9973 |      1.284   |    .855862   |    .641897!!! |
 resulting tolerance     0.9999 |      1.712   |      1.141   |    .855862    |
 ------------------------------------------------------------------------------
 STATISTICAL calculation.          Classical value of the tolerance  =  .855862
```

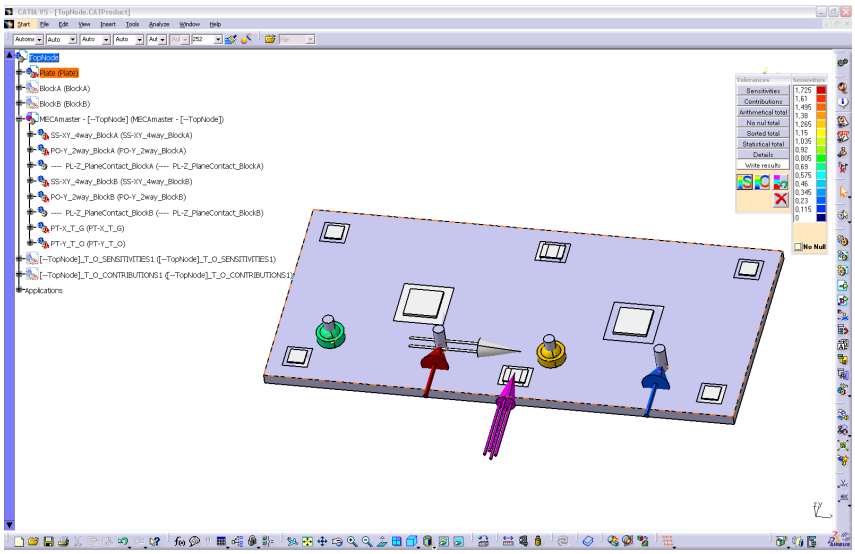

Figure 2.10: *MECAmaster calculation results for the offset tolerance T_O as text visualization including the complete 3D stack (top) and the corresponding graphical visualization of the weighted influence tolerances within the GUI (bottom)*

Upon user request, the input data can be exported to a text-based m_m-file. The key elements of such m_m-files are data blocks. All elements in the MECAmaster simulation are represented by one data block in the m_m-file. The GUI shown in figure 2.7 represents the editor window for a 'point contact', and the code fragment shown in figure 2.11 represents the corresponding data block within a m_m-file.

```
****  DATA number      2
|--------.-.------|--------.-.------|--------.-.------
 Linkage type .......... POINT CONTACT
       between ......... BlockA
       and ............. Plate
 Point Contact name .... 2way_BlockA
 Point of contact ......        -100.000000          0.000000         30.000000
 Direct. normal to plane           0.000000          1.000000          0.000000
 Precis. of Linkage ....          0.00000000        0.10000000        0.40000000
 !complements !.........
 ! 1s tol inf !......... Datum B
 ! 2n tol inf !......... Game
 ! 3r tol inf !......... Positional Tolerance
 !   criteria !.........
 !       path !.........
```

Figure 2.11: *MECAmaster m_m-file code for a 'point contact' linkage element corresponding to the CATIA-based GUI shown in fig. 2.7*

2.3.3 Monte Carlo Simulation

An alternative to the direct linearization method for 3D tolerance analysis is computer-based tolerance simulation, where the components are virtually assembled using Monte Carlo simulation algorithms [52, 73]. The assembly is repeated thousands of times – every time with the same parts but with different tolerances according to predefined tolerance distributions, which simulates real manufacturing conditions. The tolerance targets are measured and captured for every assembly. Finally a statistical distribution of the virtually measured tolerance values is generated. A software implementation using this algorithm is 3DCS[51] [42]. The working process is the following [42]:

1. Definition of all required tolerance targets called 'measurements'.
2. Definition of the so-called assembly 'moves' including the vector-based movement direction, reflecting the installation steps[52].
3. Definition of all contributing feature tolerance zones[53] including the direction of the zone and its distribution, which usually is a $\pm 3\sigma$ Gaussian distribution.

[51] Dimensional Control Systems, Troy, Michigan (USA), see http://www.3dcs.com, accessed Jan 2012.

[52] 'Moves' are a key concept of 3DCS and are captured in a list. Step by step, the complete assembly scenario is modeled. Due to this simulation approach, the assembly sequence is explicitly modeled and thus can be changed in order to investigate the influences. There are several kinds of moves for various plane or line alignments, for the alignment of holes to holes and even special bending routines.

[53] Comparable to the input possibilities with MECAmaster, in 3DCS the tolerances contributors t_i may also be defined as independent, grouped or composite.

4. Automated Monte Carlo simulation:
   (a) Creation of component sets[54] of all assembly components.
   (b) Random selection of the components out of the sets and assembly according to the moves.
   (c) Check, whether the assembly succeeded, followed by the measurement of the variation[55].
   (d) The assembly simulation is repeated until every sample in every set has been used once.
   (e) Creation of simulation result visualizations, such as statistical results, distribution curves, or deduced non-linearized influence coefficients[56].
5. Post-processing of the simulation results including output to linked analysis tools[57].

The Monte Carlo simulation itself can last several minutes, depending on the model size. In practice, usually several simulations with different input values are performed in order to check if the modeled assembly scenario works properly. The 3DCS GUI is a tool bar in CATIA [42]. The original 3D CATIA parts are copied to the 3DCS part stack using simplified geometry representations and the entire part is displaced during the assembly process. Every change of the CATIA parts has to be updated into the 3DCS environment. The 3DCS data is saved in the CATIA file format.

2.3.4 Comparison

The creation time for MECAmaster models is usually fast and modifications can be implemented easily. However, as soon as the linkage system requirements get complex, for instance with interrelations between local datums or with huge influence of the assembly sequence, the manipulation time of the model increases. This may even lead to the necessity to create independent models for single tolerance targets and for each installation sequence variant. In such cases, modeling automatization is helpful.

As all calculations are based on independence between the contributing tolerances, no statements about depending target tolerances can be made. For example, if two gaps are measured, it cannot be stated if both tolerances are exceeded, or if a bad one on one side is associated with a good one on the other side.

If the attachment concept is hyperstatic, modeling workarounds have to be set up. Coupling with FEM in order to analyze the influence of deformation due to hyperstatic conditions is complex and time consuming and there is limited industrial experience. Within MECAmaster, load analysis algorithms for isostatic load cases are implemented [113].

The major advantage of simulation tools like 3DCS is that the modeling flexibility of the assembly moves is larger, as the simplifying linearization steps of kinematic tools do not apply, offering the possibility to model more exactly and in more detail. Of course, this may also lead to higher modeling time which makes the application questionable for the conceptual design phase, where fast trade studies are more needed than time-consuming precise simulation results.

The simulation of hyperstatic attachments is possible where part deformation can be estimated simultaneous to tolerance simulation using a FE mesh-based geometry deformation theory[58]. This, however, requires good knowledge about the deformation behavior of the part and about deformation calculations in general. Additionally, the preprocessing becomes more and more difficult and time-consuming.

[54] The tolerance distribution of every set of each part follows the previously defined distribution. In industrial practice, usually between 5000 and 15000 assemblies are simulated, if $\pm 3\sigma$ results are favored.

[55] All variation 'measured' by simulation represents statistical results. No exact worst-case calculation is possible using Monte Carlo simulation techniques. Workarounds are implemented in 3DCS, but the user has to be clear that 'simulation' is different to (linearized) 'analytical calculation'.

[56] Unlike in MECAmaster, in 3DCS the influence coefficients can only be calculated with a separate so-called 'high-low-mean' simulation ('sensitivity analysis') [42], which needs all contribution tolerances to be specified before.

[57] E.g., 3DCS has been complemented with a tolerance-cost calculation method recently [41].

[58] Only geometry deformation is considered, loads and stress behavior cannot be modeled.

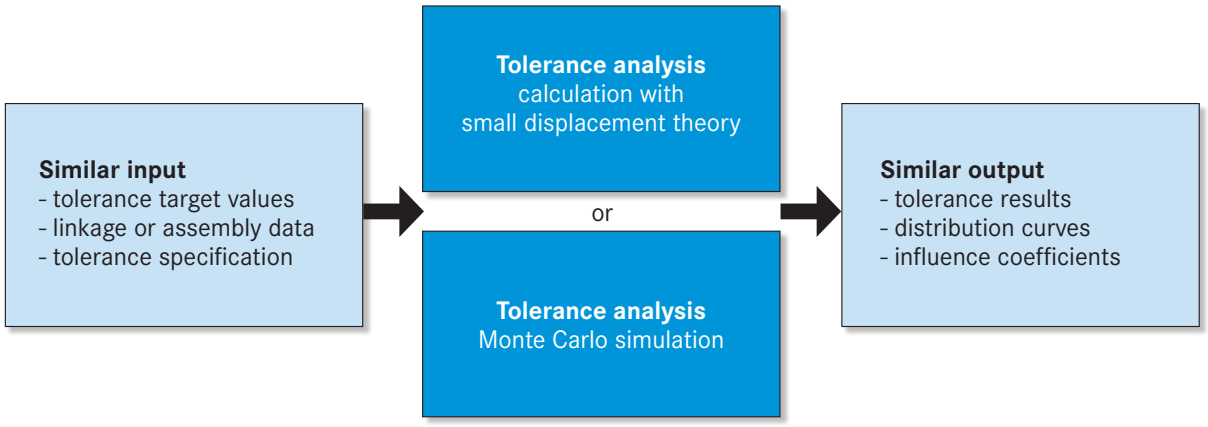

Figure 2.12: *Theoretical interchangeability of CAT software within tolerance analysis processes*

To summarize the following statements can be made:

- In general Monte Carlo simulation algorithms resemble the real assembly processes better than kinematic methods using linearization methods, especially where the installation sequence has high influence on the tolerance results.
- The precision of Monte Carlo simulation demands modeling and simulation time. For simpler tolerancing tasks with mostly kinematic interface functions, the kinematic modeling is usually faster and leaner than simulation approaches.
- It is important to be familiar with the simplification assumptions in MECAmaster, especially when workarounds are necessary for linkage and assembly sequence modeling. On the other hand, understanding the complex simulation assumptions in 3DCS is also difficult. The risk for 'believing in the tool' instead of 'knowing what happens' is given for both methods.
- Concerning accuracy in principle no significant differences have to be expected [52]. SALOMONS [137] and PRISCO [126] provide more theoretical comparisons and come to similar conclusions.
- Input and output data *types* of both tool families are comparable and combined or interchanging applications are possible (fig. 2.12). The input data *files* and the data stowage are different, so exchange needs data or model transformations.
- For applications in the aircraft cabin concept phase with mostly kinematic linkages and the need for simplification assumptions, kinematic tools like MECAmaster are preferable. The strength of simulation software like 3DCS lies in a later application for detailed design and manufacturing.

2.4 Research on Tolerance Analysis

WHITNEY and MANTRIPRAGADA extend tolerance analysis to assembly analysis [104, 170]. The concepts of mate or contact joints respectively as discussed in chapter 1 offer to anticipate manufacturing constraints, which can be visualized by so-called 'datum flow chain' diagrams [104]. These diagrams indicate, which component of an assembly depends on which other component concerning the kinematic degrees of freedom in between[59]. The classification of assemblies into type-I and type-II can help to understand, how the functional design of physical interfaces influences manufacturing processes.

KOLLER contributes on the incorporation of contact forces into tolerance calculation models [84]. MEERKAMM and HOCHMUTH focus on the role of tolerances in the product development process and on the development of integrated product development systems including links between CAT and FEM or between CAT and quality methods [71, 72, 114]. LUSTIG sets focus on the integration of tolerancing and deformation analysis [100]. GERMER [54, 55] presents an interdisciplinary approach to embed tolerance management activities in conceptual product development. However, the back flow from research to industrial application usually takes its time. Despite the fact that the tolerance-deformation coupling is well-described in academic terms, it is still common industrial practice to couple CAD results and FEM results manually.

MARGUET et al. list special aircraft tolerancing requirements [105]. On this basis, a method and a tool called GAIA (Graphical Analysis of Interfaces for Assemblies) have been developed [49] for graphical analysis purposes. The research aim is to represent the airframe production processes based on functions, tolerance specification and production frame conditions [13, 34, 50, 106], following the GASAP ('geometry as soon as possible' [12]) principle. Key Characteristics (KC) according to EN 9100:2009 are regarded as functional requirements and thus ask for early embedment in the design processes[60].

SUH describes an axiomatic approach to design [99, 157, 158]. He develops four spaces or domains for customer needs, functional requirements, design parameters and process variables. For each domain, a matrix contains the corresponding parameters or variables, which need to be expressed in a formal way. To that effect, designing is considered to be mapping or transformations between these matrices. So-called design axioms, which propose beneficial design decisions in the context of coupled design aspects can be consulted for robust design under consideration of design constraints. For instance, [4] shows a basic application example for the method. RUDOLPH sets the axiomatic design approach into the context of the evaluation hypothesis and similarity mechanics [131] and describes basic mathematical formulations of the so-called design axioms [132].

As already mentioned before, HILLSTRÖM works with a concept of 'interfaces' [70] looking at spatial and positional aspects. Further research has been conducted in the contact field of axiomatic analysis and tolerance management by JOHANNESSON, who provides contributions extending the technical understanding of functional coupling analysis [77, 78]. SÖDERBERG proposes a method to link tolerance management with quality and manufacturing cost analysis [147, 149]. From an axiomatic point of view, these research contributions represent methodological linkages between the functional domain (quality), the physical domain (tolerances) and the industrialization or procedural domain (manufacturing cost model). Furthermore, they investigate the link between functional decomposition and the resulting coupled tolerance chains[61] axiomatically [79, 148, 150]. They propose a graph visualization for the detection of unwanted tolerance chains, which are couplings in the sense of the axiomatic design theory [157], leading to a method of achieving robust design from a tolerancing point of view.

[59] Compare also with the abstract MECAmaster visualization of 'kinematic linkages' shown in fig. 2.9.

[60] Section 2.6 will show how this is done for a cabin tolerance management process.

[61] As will be shown later in chapter 5 using design languages, coupled tolerance chains can be identified as closed loops in the design graph. From this point of view, a tolerance calculation checks for the repercussions of coupled design aspects.

2.5 Tolerance Synthesis

Manifold contributions were made to investigate correlations between tolerance management including tolerance synthesis, manufacturing processes and capabilities or between quality and cost and to reflect these interdependencies within product development methods and process frameworks. As mentioned before, tolerance synthesis herein means to 'synthesize' or to 'anticipate' single tolerance values empirically or (semi-)analytically in order to enable assembly tolerance analyses subsequently. For instance, equation I can be used iteratively for the tolerance synthesis of the individual tolerance contributors t_i, if they are constituted by sub-assemblies and if the contributing sub-assemblies tolerances are known.

For the most part, tolerance synthesis is a very special technical problem associated with the individual products and their manufacturing processes. The already mentioned standard ISO 2768-1:1989 for general tolerances and some further standards – mostly for metal processing – constitute basic solutions for tolerance synthesis problems. It is also common practice to use experience or 'best engineering guess' values and heuristics, in particular for conceptual design. For instance, FARMER proposes an empirical formula to estimate single part manufacturing tolerances for a given dimensional value D based on a tolerance grade factor IT representing the chosen manufacturing technology [51]:

$$t = \left(0.45 \cdot \sqrt[3]{D} + 0.001 \cdot D\right) \cdot 10^{\left(\frac{IT-16}{5}\right)} \tag{XVI}$$

This procedure is closely related to the aforementioned semi-empirical formulas and correlations of preliminary aircraft design methods and shows exemplarily the need for flexible tolerance synthesis methods for practical applications.

On one hand, generalization is difficult for complex synthesis problems with multi-disciplinary design influences. On the other hand, experience from the aerospace sector tells that rather internal production capability documents and databases based on experience and measurement are used for tolerance synthesis methods. Published standardization for special tolerance synthesis methods are often not intended in order to protect the industrial know-how.

Among the available more recent publications with a focus on tolerance synthesis, a clear tendency towards expert systems and multi-disciplinary KBE methods is observable. WILHELM [171] and HAYES [65] propose a computer method framework for tolerance synthesis integrating geometrical, functional and knowledge-based aspects. SCHEER [140] and WITTMANN [172] present a feature-based approach using a common database containing economical, measurement engineering, manufacturing and functional data on a component level. The associated method proposes a control cycle to ensure the adequate fulfillment of requirements coming from these disciplines.

MANARVI proposes using two geometry-based (CAD-based) databases: one containing product design and development data, the other containing methods and techniques information [102]. Based on work about FE analysis [101], links between tolerances and deformation are incorporated. The databases are embedded in an integrated tolerance synthesis working process for component design. STEINBICHLER [156] shows a recent special contribution from the field of molded-injection part design.

Already in 1973, SPOTTS [151] introduced an allocation method for tolerances to minimize manufacturing cost. In the late 1970s, the 'loss function' of TAGUCHI[62] came up, which now has many applications [73, 74]. Altogether, they describe methods for links between tolerance synthesis and quality evaluation. PARK introduces a method for robust design in the same context [118], while CHOUDRI describes the introduction of methods for 'Design for Six Sigma' (DFSS) into the aerospace sector [25]. MANNEWITZ provides investigations about the link between computer-aided design (CAD) and computer-aided quality assurance (CAQ) methods [103], where statistical tolerancing methods are applied to enable cost reduction.

[62] E.g., KLEIN [81, 82] can be consulted.

2.6 Cabin Tolerance Management

Neither the mentioned standards nor the literature provide universal procedural advice about how to involve tolerance management processes in industrial product development and processes. Like tolerance synthesis, engineering processes for tolerancing – especially within complex concept and development projects – strongly depend on the industrial branch, on the companies and on the educational background including design traditions. Experience shows, that larger development companies within the aerospace and automotive sector tend to write individual engineering procedures in order to control tolerance work processes which suit the applied methods well.

For the cabin concept and design phases, tolerancing engineers do not only specify or calculate tolerances, but 'manage' them paying respect to product-related and to industrial frame conditions. The aim of these cabin tolerance management activities is to define the repercussions of tolerances already in the conceptual phase of the aircraft, rather than during the manufacturing phase – and to establish the links to the neighboring design and manufacturing engineering groups like to aircraft structure and bracket tolerancing teams. The focus is to enable suitable manufacturing processes for the FAL and to prevent safety or quality issues during operation already in the design phase.

Cabin Tolerance Management Working Process

The cabin tolerance management activities stick to the following working process (see fig. 2.13):

1. Define performance key characteristics (PKCs) based on function and appearance requirements.
2. Define attachment system of the cabin modules and potential installation PKC tolerances.
3. Define cabin module datums and specify manufacturing key characteristics (MKCs).
4. Define cabin integration datums and corresponding aircraft assembly key characteristics (AKCs).
5. Calculate PKCs.
6. Discuss results with involved design and product teams and iterate with adequate corrective means.

The PKCs are product architecture-related assembly tolerance requirements T_j[63] and have to be seen as architectural design metrics. For cabin tolerancing, a PKC usually represents an acceptance criterion, so PKCs have to be physically measurable. They originate in safety or certification requirements (e.g., minimum aisle width), appearance quality requirements (e.g., gap size or parallelism) or functional requirements (e.g., clearance between movable parts). AKC and MKC tolerance specifications are subordinated tolerance requirements which serve the purpose of calculating PKC tolerances. They are the tolerance contributors t_i in 1D or 3D tolerance calculations, which the PKCs are calculated with.

Due to the long *design phase* of aircraft[64] there is the approach to conduct several tolerance analysis loops for different maturity levels of the design, ending when cabin module and component production starts. In the first loop conducted at the beginning of the *conceptual design phase*, the major PKCs are captured and initial datum definitions are set up. At this stage, AKCs and MKCs are usually based on an engineering best guess, heuristics or – if available – on history data and are communicated using spreadsheet-based KC-lists. The only prerequisites required are rough geometry definitions and some design and manufacturing input for the cabin modules respectively the aircraft structure and brackets along with initial manufacturing tolerance estimations. The initial tolerance calculations are meant to indicate whether the chosen architecture and design are capable of fulfilling the focused quality requirements and of ensuring robust and convenient manufacturing processes, using the PKCs as design metrics.

[63] See subsections 2.3.1 and 2.3.2.

[64] A complete design phase may take five up to seven years.

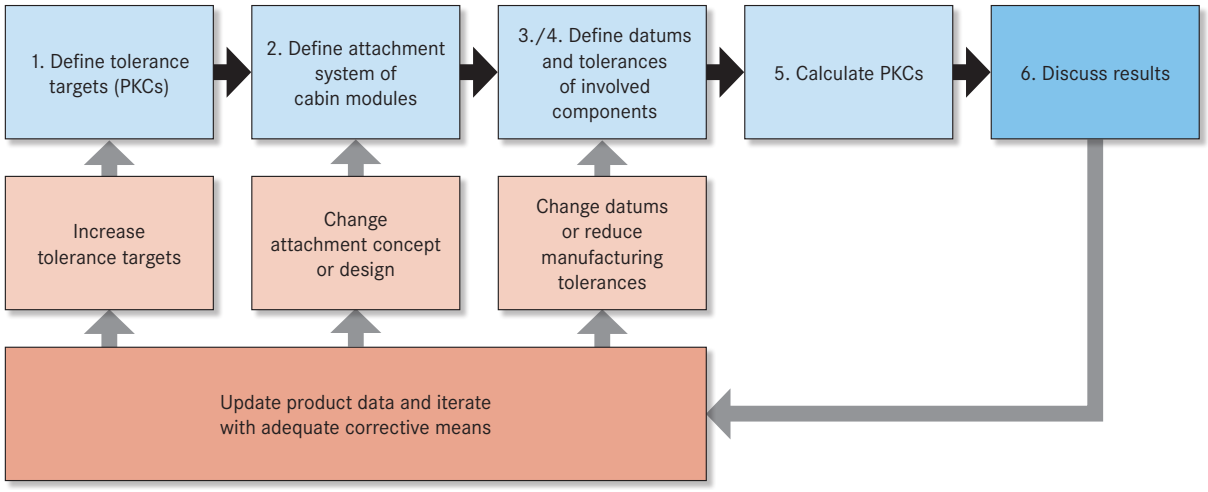

Figure 2.13: *Aircraft cabin tolerancing working process*

With the subsequent loop during the *embodiment design phase*, the number of the calculated PKCs increases along with continuous updates of the AKC and MKC tolerance values. If design changes develop, the implications on the PKCs, the attachment systems, the datum systems and the tolerances are propagated and corrective means are initiated, as figure 2.13 shows. At the end of the *detailed design phase*, just before production starts, the last analysis loop is accomplished and the tolerance requirements are finally frozen within the technical specification documents going to the manufacturing suppliers.

If a need for change evolves due to manufacturing, time or cost implications during the *production phase*, the working procedure is similar, except for the reasonable approach to limit the repercussions locally. Where possible, production data is used for tolerance calculation updates. However, unlike for the automotive industry, there is limited statistical data available for aircraft cabin modules and components due to the aforementioned small batch sizes and due to the high rate of manual work.

Corrective Means for Tolerance Problems

Within all iterative loops, the following corrective means can be used to reach the tolerances:

- **Change datum system:** The earlier a datum system is changed, the smaller the implications are. In the concept phase, changing datums is an appropriate instrument to optimize the overall product performance concerning tolerance aspects. As soon as manufacturing processes are set up, it becomes very expensive, since datum changes usually come along with the change of manufacturing processes and equipment. It is helpful to define datum systems providing high accuracy where it is needed, 'shifting' geometrical variation to non-sensitive areas. The integration datum systems have a high influence on the final accuracy of the interface points for FAL assembly processes.

- **Change attachment system, assembly sequence or entire design:** For a design or assembly sequence change, the same factors apply as for a change of the datums system. On top of that, there are engineering change efforts for the product design data. Nevertheless, a design change can still be a relatively cheap method compared to continuous quality or manufacturing problems due to an unfavorable design. Again, necessary design changes should be identified as soon as possible to limit the cost impact for development efforts or even for manufacturing. When adapting interfaces and the assembly sequence, it is important to pay respect to the various interrelations of the multiple interface points and the corresponding tolerances, as well as to consider the repercussions for all involved engineering disciplines.

– **Reduce manufacturing tolerances (AKCs and MKCs):** The seemingly simplest method to achieve a tolerance target (PKC) is to reduce the contributing manufacturing tolerances according to the 1D or 3D tolerance stack. From a manufacturing point of view, this is the most expensive one yet. If the reduced tolerances are manufacturable at all, they are usually much more expensive[65]. Especially when dealing with all-new products, it is additionally very risky to 'bet' on potential manufacturing capability improvements in the field of tolerancing.

– **Increase specified tolerance targets (PKCs):** This 'trivial' change is always the simplest method to achieve the target – but with the most repercussions on product performance. The implications on quality and installation aspects need to be clear when increasing the tolerance specification of a functional feature. In particular this can mean pushing the risk away from component manufacturing into the FAL[66] (for installation-related tolerances) and could put risks on customer satisfaction[67] (for appearance quality-related tolerances). However, during the late development phase, this is often the only way to proceed without changing the entire design or assembly concept.

Specific Cabin Tolerancing Aspects

It is thus the task of the tolerancing experts to discuss the above named corrective means with all disciplines involved and to push for the best one for the product – considering the whole product life cycle, in particular including component manufacturing, assembly and customer quality concerns. Cabin module installation in the FAL is completely different from structure component joints on the fuselage section level (see fig. 2.14). Conceptual tolerancing aspects of the latter have already been outlined by MARGUET [105]. For dedicated conceptual aircraft cabin tolerancing, however, no scientific research work has been done up to now.

	Cabin Tolerancing	Structure Tolerancing
Geometry	Mostly simple geometries: cabin modules of either planar shape (panels) or cuboid shape (stowage bins, monuments)	Frames, panels, complex frameworks and substructures incl. brackets
Types of interfaces and assembly processes	Attachment principle with brackets and dedicated interface points, clear installation sequence, manual installation	Riveting with overlapping, joining and assembling processes, machining, automated processes where possible
Installation time requirements	Very strict, due to high work load in 'expensive' FAL	Focus on standardization and prevention of cabin customization impacts
Datum systems	Cabin modules with repetitive datum systems. Many group tolerances and common zones over large areas for the bracket interface points	Complex section-wise datuming with the focus on fuselage integration

Figure 2.14: *Comparison between specific cabin and structure tolerancing aspects*

Due to the enormous number of KCs and due to the risk of overwhelming design changes, the tolerancing processes needs to be flexible and adaptable to individual problems. Sometimes, a tolerance problem

[65] The above mentioned quality-related publications discuss this problem in detail. According to KLEIN [82], a very rough rule of thumb says that a reduction of a tolerance by half the value leads to an increase of manufacturing cost by the factor four.

[66] For instance, if a pin-slot interface tolerance target is increased beyond the compensation capability of the slot, there is the risk that installation fails. Thereby the apparently cheaper component manufacturing (due to the increased allowed deviation) may lead to much more expensive assembly problems in FAL.

[67] If *designers* accept a larger gap tolerance between to cabin modules, it does not necessarily mean that the *customer* also likes this solution.

with multiple repercussions on design or even architecture aspects like time, cost, quality or performance cannot be calculated completely due to time restrictions. On one hand, this means that tolerance engineers need to be able to assess the problem based on experience and on heuristics. On the other hand, there is the chance to deploy automatization software to speed up tolerance analysis, KC-list compilation and the analysis model creation in particular. Especially in the context of modern engineering methods, the latter one becomes more and more interesting for industrial reality, which is also reflected in the recent development trends of commercial CAT software, as described above.

A current tendency goes into the direction of involving embedded tolerancing methods early in the conceptual design phase for cabin design, architecture and integration. The aim is to prepare and evaluate new technical cabin architecture scenarios (research and technology). However, while the role of cabin tolerance management during the *embodiment design* and the *detail design phases* [117] is clear as described above, there is only limited methodological experience concerning how to involve tolerancing methods early in the *conceptual design phase* of cabin architectures.

Chapter 3

Physical Aspects of Cabin Architectures using Tolerancing Methods

3.1 Problem Description

The overview provided by the two previous chapters showed that several methods are available, which can be adapted and applied for the needs of physical architecture analysis during the early conceptual cabin design phase. However, there are open questions and missing links at crucial points, which prevent fast, brief and target-oriented analysis of technical scenarios. These can be expanded into the following two questions A) and B):

A) Which methods and which models are needed for the evaluation of physical cabin architectures during the early conceptual design phase and what is the role of tolerancing in this context?

In order to be able to analyze physical aspects of aircraft cabin architectures, it is mandatory to define which methods need to be considered and to show, how conceptual tolerancing can be embodied. Traditionally, there has been limited perception for dedicated tolerance management methods in design theories. For instance, PAHL and BEITZ [117] do not define these methods as dedicated engineering methods, particularly not for conceptual design. In addition, industrial experience shows that tolerances are often considered as manufacturing-related problems rather than as conceptual design parameters with influence on the whole architecture.

But especially technical product architectures, which are widely influenced by mechanical interfaces – such as the brackets between the cabin modules and the aircraft structure – need methods with a focus on interface design. In particular, it is required to support architectures and design for cabin and fuselage in parallel instead of a sequential integration of cabin after the fuselage architectures are frozen. To close this gap, tolerancing could be involved early in the concept phase of product development, since tolerancing methods can bridge between the interface architecture and manufacturing-related or quality-related aspects.

B) How can the application of these methods be implemented in a fast and pragmatic as well as a scientifically sound way?

There is the need for pragmatic implementations of these methods within a supportive software framework in order to offer an alternative to intuition-based design heuristics for architecture design. In particular, there is a need for consistent and time-effective conceptual multi-domain trade studies.

A look into the status of research about dedicated tolerancing methods given with chapter 2 shows, that CAT software is important for the given analysis task, but cannot answer the questions by itself. The work on graphical interface analysis done by MARGUET et al. [12, 49, 105] with regard to link tolerancing with assembly process modeling heads in the right direction. However, it seems that the presented method cannot display more holistic design aspects beyond functional tolerances, and the extension to such representations is difficult and not intended. A root cause for this is the core principle GASAP[68], which stands for a strong adherence to a geometry paradigm[69]. The approaches from JOHANNESSON, HILLSTRÖM and SÖDERBERG [70, 79, 147] can be helpful support methods to analyze the role of functional tolerances. The analogy to design graphs evolving from design languages is remarkable and seems to open a research space for axiomatic tolerancing with design languages. The evolving tool [150], however, does not aim at representing arbitrary multi-disciplinary integration models beyond the axiomatic tolerancing aspects. The mentioned tolerance synthesis and quality assessment methods focus on the development of smaller products and thus the scalability to complex aircraft cabin is limited.

Outside the field of tolerancing, a shortcoming of the traditional methods for physical integration discussed in section 1.2 is that they are not flexible enough to reflect the required working or analysis

[68] GASAP stands for 'geometry as soon as possible' [12], see section 2.4.

[69] In particular, the 'CAD paradigm' [16, 134] expresses the dominant role of geometry in traditional design processes and must diminish since more and more non-geometrical features start to drive the design processes.

processes for cabin architectures. CAD and CAD-related software implementations usually adhere to the geometry paradigm [97]. Moreover the modeling principle of preliminary aircraft design methods is usually geometry-based and uses geometry-focused abstractions of the aircraft [44, 91, 109, 125, 145]. For more holistic architecture analysis, it is required to overcome the shortcomings of single engineering disciplines with their individual method and model concept constraints [93].

GÖPFERT outlines strategies to implement development methods into software and spans a bridge to MBE approaches [58]. The required links to tolerancing as well as to manufacturing and industrialization aspects can be incorporated into models for MBE, if the modeling philosophy is open to implementing additional aspects. However, there is a risk of these methods 'overtaking' the users, as the modeling-related aspects can be very sophisticated and abstract. Additionally, the more complex data models and analysis methods get, the greater their implications for design processes are. A help can be to combine MBE along with methods which support knowledge-reuse, and which can be applied 'intuitively' for physical design engineers.

3.2 Solution Approach Overview

In the following, the two working hypotheses of this dissertation are unfolded:

I) Tolerancing constitutes an 'intersecting set' among the various analysis models and methods which need to be involved, and therefore can be used as a key to analyze the mechanical integration aspects of cabin design and architectures (first hypothesis of this dissertation).

It is necessary to describe an abstract *scope of modeling*[70] which needs to be developed 'around the modeling concept of tolerancing'. Based on this, a specific multi-domain meta language [134] has to be deployed, for which the *scope of modeling* contains analysis criteria and the corresponding analysis methods for the special application case of conceptual cabin architecture studies. In the center of this abstract model, concepts for cabin modules, for spatial and for mechanical interfaces need to be developed.

As a side aspect, it is necessary to sketch links between the scope of modeling for the requested analysis of physical architectures and the models of further aspects beyond this initial scope, such as load analysis or functional cabin and systems integration – the latter one in particular to prevent a irreversible decoupling of ATA25-related and systems-related integration aspects.

II) An approach using executable design languages can be used to support the model-based creation and the analysis of the involved product data (second hypothesis of this dissertation).

Hereby, the task should *not* aim at the development of 'a generic design method', but rather on a pragmatic approach for the particular context of physical integration aspects of aircraft cabin design and architecture. It has to be shown, that an approach using rule-based *cabin design languages* (CDL) following a model-based paradigm [134] can provide answers to the pragmatic questions, such as how to create and how to analyze the data models, and how to implement the proposed approach in an industrial context. It has to be demonstrated, how the CDL offer the user to focus on the rule-based creation and modification of digital product data and on the formal pragmatic scenario analysis and comparison as part of a complex holistic evaluation task, rather than on the manual creation of formal models, or on syntactic or semantic data checks.

[70] In the context of this thesis, the term 'scope of modeling' is intended to indicate that at first analysis methods along with their corresponding abstract modeling principles [155] have to be identified constituting an abstract modeling space, which can later provide explicit vocabulary [136] for design languages. In particular, the step to roughly outline the scope of modeling for physical cabin architectures on an abstract schematic level (discussed in section 4.1) needs to be differentiated from the detailed implementation-focused definition of the cabin design language class diagram (discussed in section 4.2).

Although recently the link between design languages and digital factory models came into the focus [10], until now only little research has been conducted concerning links to tolerancing activities [93]. However, tolerance target definitions, attachment system definitions, datum definitions, tolerance definition or even tolerance synthesis as well as the actual tolerance calculation (steps 1 through 5 according to fig. 2.13) are promising candidates for an implementation approach with the CDL. The principle of semi-analytical and empirical methods as used for preliminary aircraft design can be a helpful contributor for rule-based design parameter synthesis such as for AKC and MKC tolerances.

3.3 Structure of the Thesis

To develop the concept of the *cabin design languages* (CDL) within chapter 4, section 4.1 at first provides the mentioned *scope of modeling* outlining those analysis methods that need to be involved (fig. 3.1). On the basis of this, an UML-based *class diagram* is developed within section 4.2, which contains the CDL vocabulary. With section 4.3, a set of *architecture analysis parameters* (AAPs) for CDL models are presented, and section 4.4 closes the more theoretical aspects with the description of a framework for implementations into software.

In chapter 5 the implementation of the CDL within the DesignCompiler 43 environment is discussed. As mentioned, until now there is no software interface between design languages and tolerancing software. To compensate for this shortcoming, a software interface is developed along with a set of plugins for the CDL-specific implementations, which is documented in section 5.1. Section 5.2 introduces a use case for the CDL and section 5.3 demonstrates how several trade studies within this use case can be executed using graph-based executable design languages.

Chapter 6 comprises the results and discusses them in the context of the two mentioned hypotheses. More specifically, section 6.1 focuses on the technical results of the use case and section 6.2 on the methodological aspects. With section 6.3, implications for working processes in the industrial context are considered.

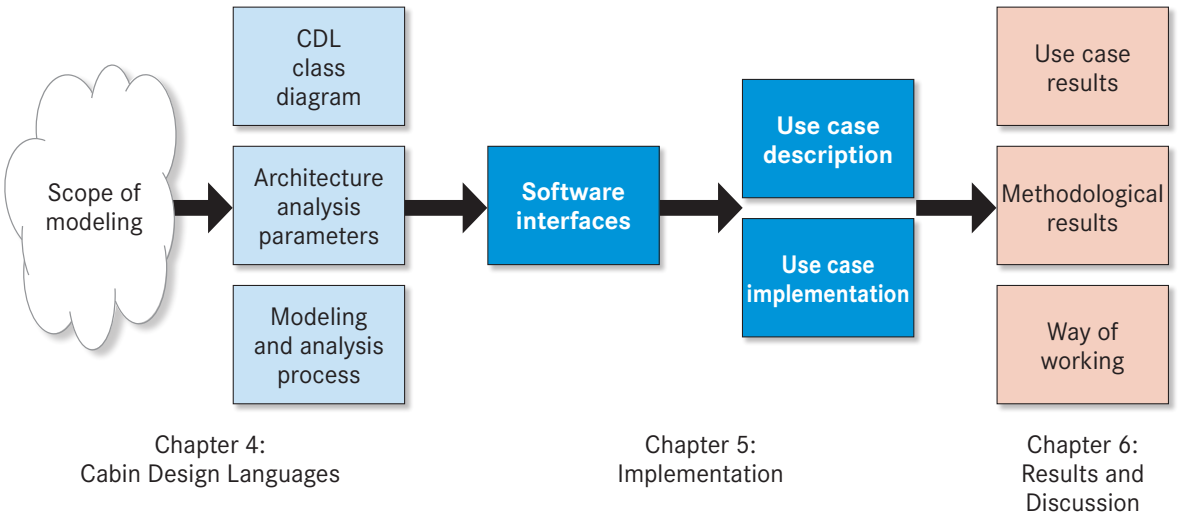

Figure 3.1: *Structure of this dissertation*

Chapter 4

Cabin Design Languages (CDL)

4.1 Scope of Modeling

As expressed in chapter 3, cabin design and architecture analysis is a multi-domain problem [93] and can benefit from a supporting multi-domain meta language [134] like a design language following a model-based paradigm. In order to describe the demanded *scope of modeling* for conceptual design purposes of physical cabin architectures, the following aspects can be outlined[71]:

- *What needs to be modeled:* model of cabin modules or system components, of interfaces and of adjacent structure components
- *When is it modeled:* after complete installation in FAL, ready for flight[72]
- *For whom is it modeled:* for cabin architects
- *For which purpose is it modeled:* to support the analysis, comparison and evaluation of different technical scenarios

Consequently, the investigated scope of modeling is considered the accumulation of those analysis methods [134] with their corresponding modeling concepts which play an important role for the outlined conceptual cabin architecture analysis task and thus should contribute to the evolving cabin design languages.

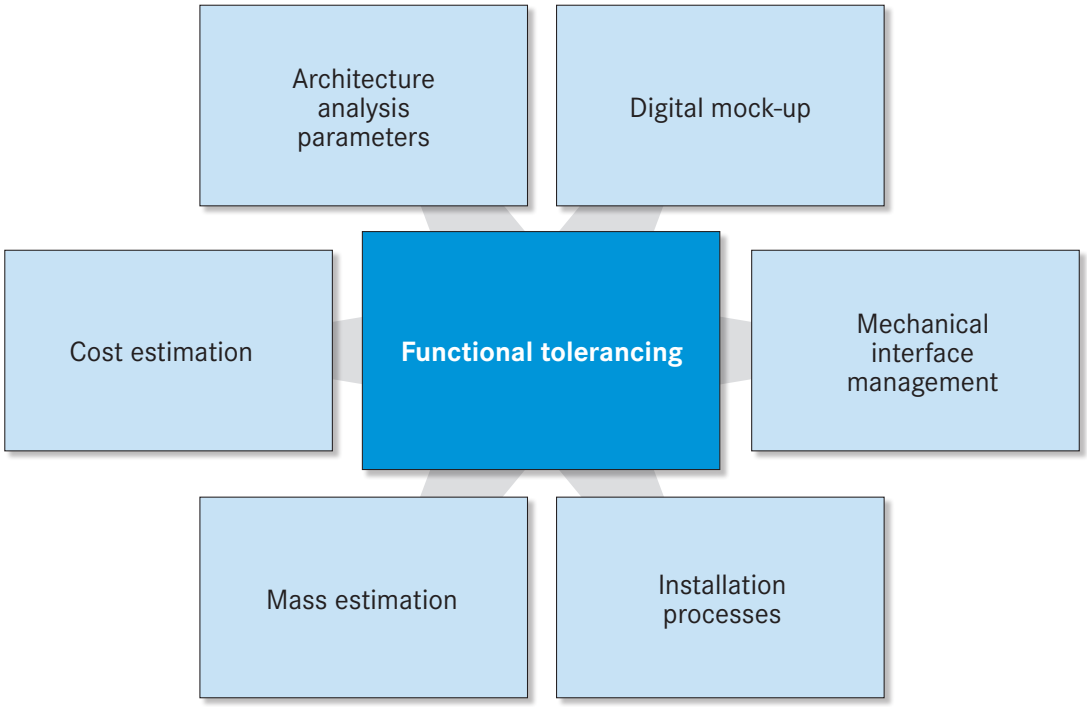

Figure 4.1: *The scope of modeling of the cabin design languages*

Figure 4.1 depicts the involved methods as well as their interdependency with the modeling concept for tolerancing. Of course, not all detail aspects of the named methods need to be considered, but those that

[71] In line with the four questions about models according to STACHOWIAK [155], see footnote 13, page 10.

[72] 'When' has to be read as the status or the system state of the cabin within the life cycle reflected by the model. In particular, the CDL models do not represent a flying aircraft, but an aircraft standing on the ground with a completely installed cabin.

interact with other methods and therefore may lead to multi-disciplinary design repercussions if changed during a technical trade study.

The *digital mock-up* (DMU) for conceptual physical architecture and integration purposes comprises the cabin modules and the cabin and aircraft system components with an appropriate level of details, schematic attachment brackets visualizations and some primary structure components. The purpose of the conceptual DMU is to prepare the spatial integration or DMU integration tasks. Spatial objects can be clustered into subcomponents. A *weight estimation* method is needed to provide the link between the cabin modules and the aircraft total weight report. According to the modeling depth as required for the DMU, the mass of the cabin modules and of the attachment brackets can be assessed using appropriate estimation methods.

The purpose of conceptual *mechanical interface management* is to get a grasp on the physical design and architecture complexity, to ensure the industrial feasibility of chosen technical concepts, to ensure a certain level of appearance quality as well as to set limits to in-flight deflections. Within the context of this thesis, tolerancing with the corresponding meta language vocabulary [134, 136] is considered as the key for conceptual mechanical interface management. Taking up the considerations from section 1.2, the vocabulary required for these tasks is:

- Functional geometry features of the cabin modules, the system components and of those aircraft structure components, which are relevant for the physical integration of the cabin and system components.
- Gaps and split lines between the cabin modules (appearance PKCs).
- Functional descriptions of attachment concepts which locate and fix the cabin modules and of potential installation PKCs.
- Definitions of the datum system of the cabin modules and of the cabin integration datum systems.
- Tolerance estimations for the considered functional geometry features.

To link aspects out of the wide field of *installation process* planning methods, the modeling objects of primary interest for cabin architecture analysis are:

- Ability to harmonize FAL and MCA manufacturing strategy on global level
- Overview of FAL installation sequence
- Type of installation per cabin module
- Number of installation steps including preliminary estimation of time per step

Based on these aspects, initial working time estimations and an overall installation sequence planning can be made. In order to be able to make statements about the detailed process length and about parallel work, very detailed information is required. As such data usually is difficult to obtain, the focus of the architectural analysis goes towards a more global anticipation of the repercussions of a chosen design and architecture for the installation processes. For conceptual *cost estimations* in the context of the given architecture analysis task, the cabin module-related manufacturing cost can be broken down as proposed in figure 1.4, chapter 1.

Using these concepts, both architecture complexity analysis using *architecture analysis parameters* (AAPs, see section 4.3) as well as tolerance calculations are possible. FEM-based load and deformation analysis, which aims at the sizing of the detailed geometry and at the analysis of the in-flight deflection behavior is not required for now. Nevertheless, except for an automated FE meshing of the cabin modules and of the structure components, the information required to launch the FEM analysis has to be part of the model.

4.2 CDL Class Diagram

Within this section, the key concepts of the afore-introduced *scope of modeling* are transferred into UML classes [128]. This follows the proposed concept of design languages with a model-based paradigm according to hypothesis II (chapter 3). At this level, the CDL class diagram does not yet comprise case-specific classes, but only generic ones in the style of a 'generic model kit'. Specific classes are intended to be defined in special case-specific class diagrams. Figure 4.2 shows the main aspects of the CLD class diagram including most class attributes and associations[73].

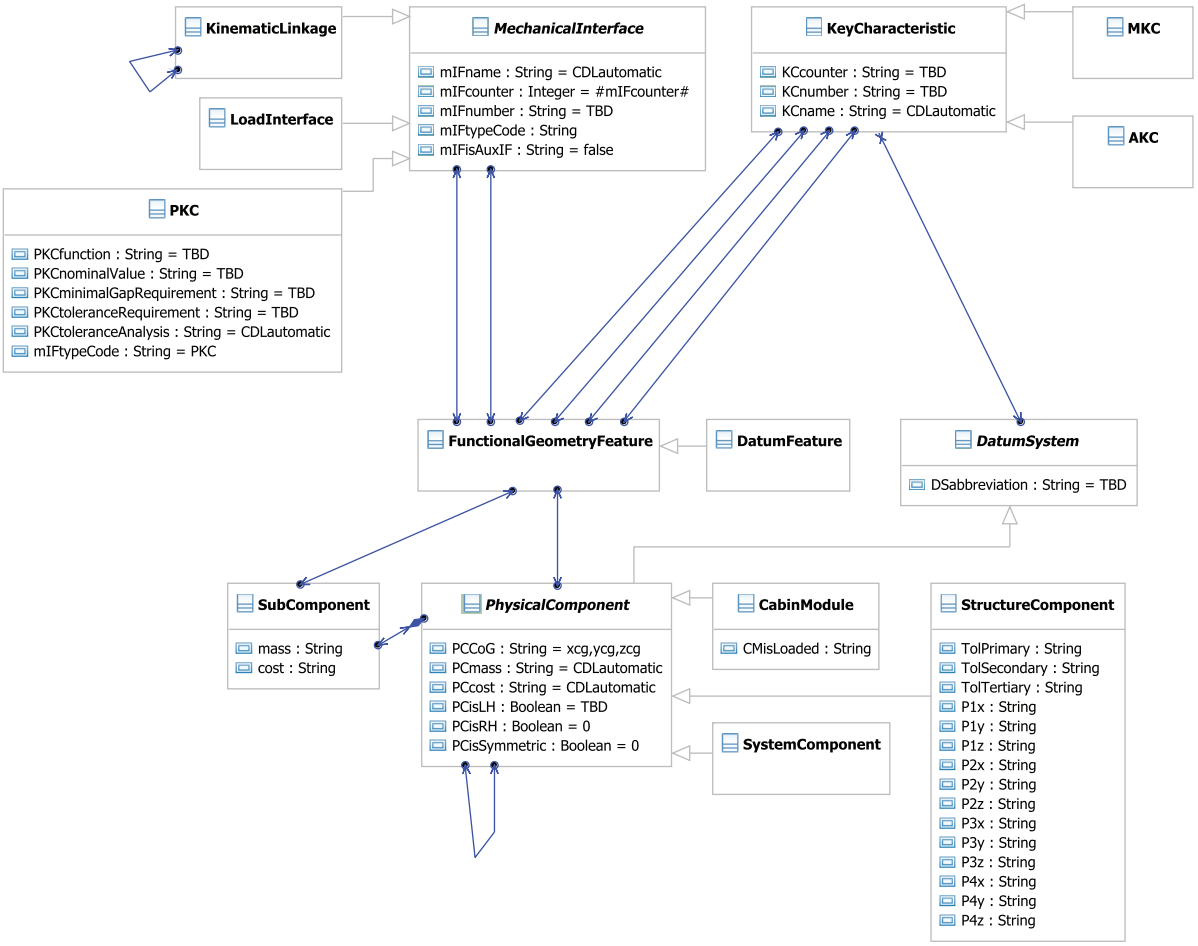

Figure 4.2: *The CDL class diagram*

In the following, the individual classes shown in fig. 4.2 will be described. The main classes are *PhysicalComponents* and *MechanicalInterfaces*. Most other classes either inherit from these two classes or are linked with them[74].

The semantic hull [135] of concrete analysis models is constituted by the CDL class diagram including inheriting case-specific classes and by the design rules considered for creating the models, which will be outlined in section 4.4.

[73] See appendix B for the full CDL class diagram including all attributes, inheritance relations and detailed explanations.

[74] The creation of a class diagram represents a major intellectual effort for which a deterministic procedure is not yet know. Therefore, the establishment of a class diagram is skill reflecting individual preferences. Its structure is a compromise between general validity, specialization for individual applications and the available implementation in software [136].

4.2.1 Physical Components, Datum Systems and Process Steps

Taking up the modular way of thinking according to MBE [136] and systems engineering [40, 117], each physical entity forming part of the aircraft can be considered a 'physical component', such as cabin modules, system components and aircraft structure components. The generic classifier *PhysicalComponent* can represent any object within the DMU. Therefore, it inherits from the abstract class *Position*, which provides attributes for Cartesian coordinates and rotation angles[75].

Along with a mass and a manufacturing cost attribute (*PCmass* and *PCcost*), further standard attributes are the center of gravity (*PCCoG*), as well as three boolean attributes to specify, if the instances are left-hand, right-hand or symmetrical within the DMU (*PCisLH*, *PCisRH*, *PCisSymmetric*). In order to model the individual design aspects for concrete cabin modules, additional attributes can be appended by creating inheriting special cabin module classes.

The CAD geometry data is not intended to be mapped on the *PhysicalComponent* classes, but on associated *SubComponent* classes. *SubComponent* instances are a model container for DMU visualization geometry along with some geometry and mass estimation parameters. The mass of a *PhysicalComponent* is calculated as the sum of the masses of all *Subcomponents* belonging to it.

For tolerancing-related and interface-related aspects, the *PhysicalComponent* class has to act as the datum system in the sense of ISO 5459:2011 for all corresponding geometry features, even if the explicit definition of the datum features is not yet established. Therefore, the *PhysicalComponent* class inherits from the abstract class *DatumSystem*, as can be seen in figure 4.2.

If *DatumFeature* objects are additionally used to model the datum system of a *PhysicalComponent* in an explicit manner, these model objects can constitute a primary, a secondary and a tertiary datum plane. In this case, it is necessary to check for the pragmatic correctness of the definitions. Completely automated datum feature definitions are neither required nor foreseen at this stage.

4.2.1.1 Cabin Modules

Within the CDL model, each individual ATA25 module gets a corresponding *CabinModule* instance (see figures 4.3 and 4.4). In addition to the attributes of the *PhysicalComponent* class, from which it inherits directly, the *CabinModules* have the boolean attribute *isLoaded*. It indicates whether a cabin module is heavy enough to make adjacent components fail due to clashes caused by in-flight or crash deflection movements. This attribute is used later to define the minimum gap sizes around the cabin module. At this stage load-stability is taken for granted for the cabin modules[76]. For the scenario-dependent definition of the architectural position within the fuselage section, individual attributes can be added by using the inheritance principle again[77].

For production planning purpose, the *CabinModule* class inherits from the *ProcessStep* class. This class provides slots for the installation order (a number saved in the slot *PSorder*), for the total installation time of the *CabinModule* and the corresponding preparation time (*PSInstallationTime* and *PSPreparationTime*)[78], the number of the FAL workers involved (*PSWorkers*), the installation place (*PSInstallationPlace* with 'FAL' as the default value for *CabinModules*), as well as the slot *PScost* for the process step cost (fig. 4.3). For cost analysis, cost synthesis functions can be added to the individual cabin module classes, which is preferably done within the case-specific class diagram.

[75] In addition to pure geometrical data, there can be case-specific architectural position and shape parameters. Such case-specific UML attributes, however, are not part of the generic CDL class diagram, but of case-specific class diagrams.

[76] The definition of the minimum gap requirements is the linking point to implement effects of dynamic deformation of the fuselage, e.g., by linking the model with FEM analysis. However, meshing data for FEM is considered more detailed design data and is not needed in the CDL model, but could be added any time after the geometry definitions (see subsection 6.2.2).

[77] For instance, an attribute could indicate the position of the frame to which a cabin module is related.

[78] This attribute stands for the time in FAL, which is needed, e.g., to carry the cabin module into the fuselage or to pre-assemble the cabin module outside the fuselage.

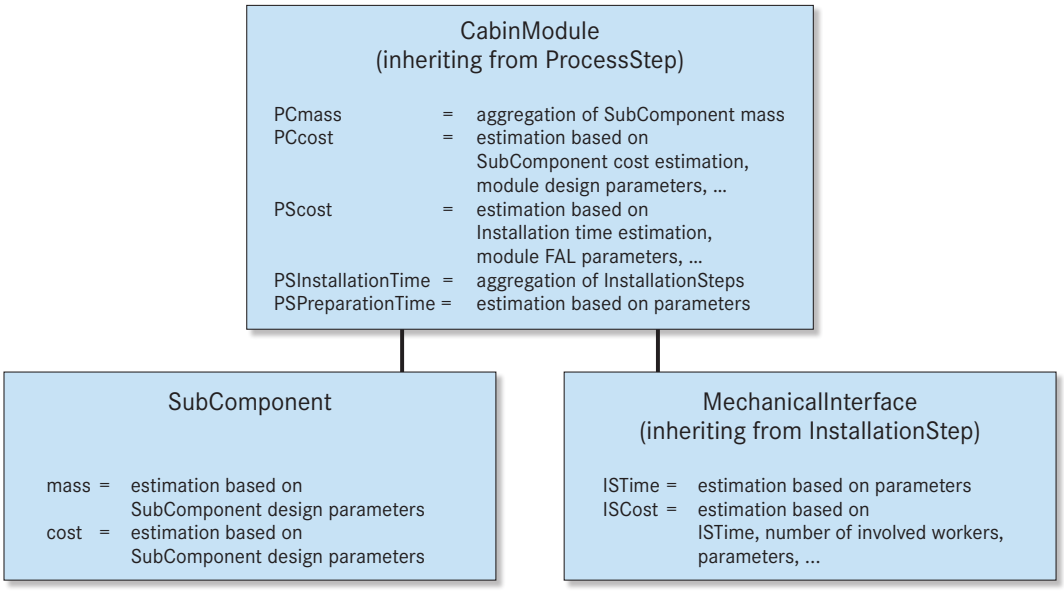

Figure 4.3: *The CabinModule class with the implementation of mass, installation time and cost synthesis on the CDL class level*

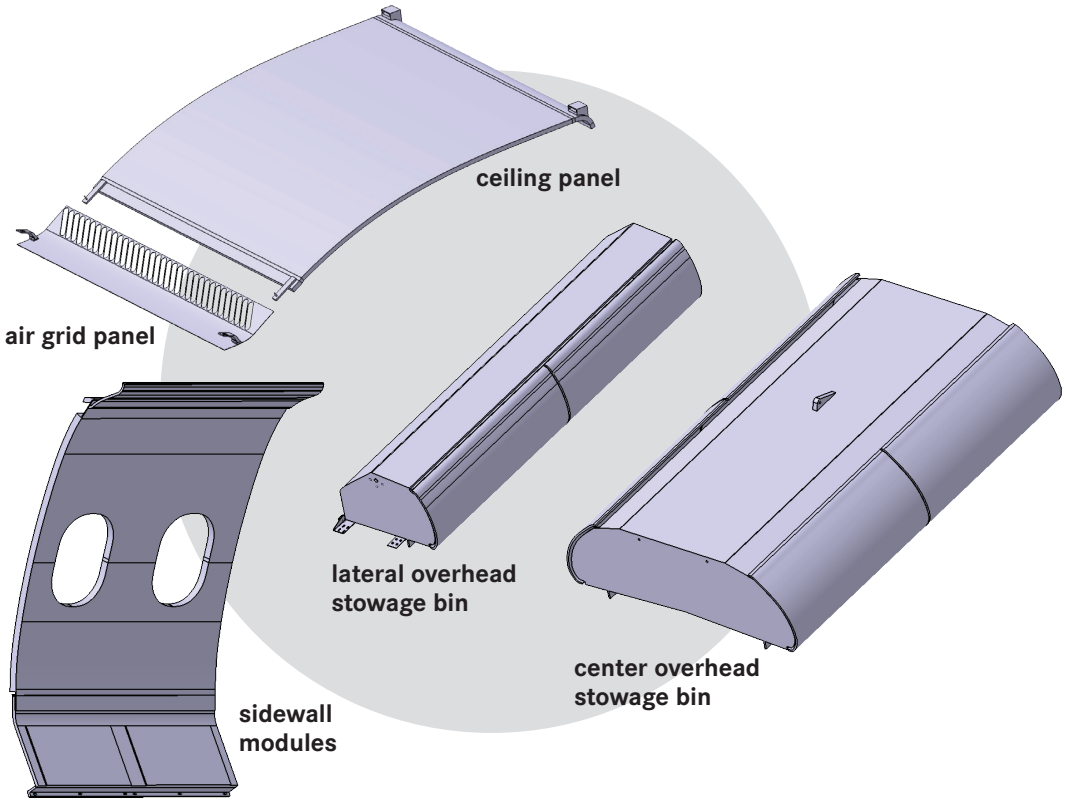

Figure 4.4: *Examples for cabin compartment (ATA25-2) modules*

4.2.1.2 System Components and Structure Components

Any part or module that does not belong directly to ATA25, but belongs to any of the aircraft or cabin systems can be specified using the classifier *SystemComponent*. It is possible to model the physical aspects of systems components in a way comparable to that used for cabin modules. The *StructureComponent* instances can be considered a placeholder for detailed aircraft structure design aspects from a DMU point of view. For tolerancing and manufacturing-related aspects, the *Structure Component* instances act as an integration datum system. They are a kind of container for *FunctionalGeometryFeatures* for interfaces to *CabinModules* or *SystemComponents* and the datum for the structure tolerances (AKCs).

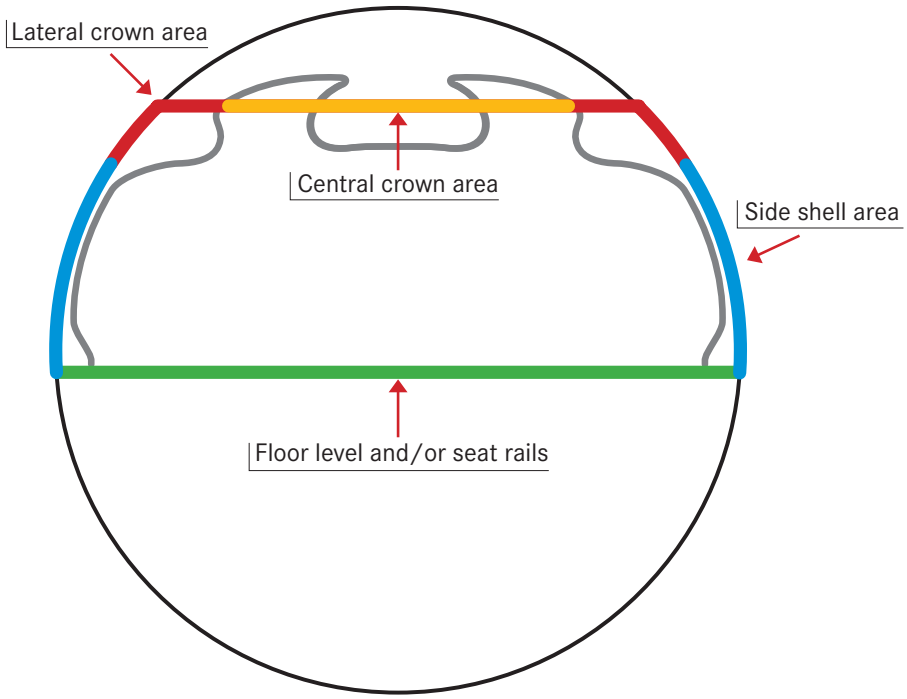

Figure 4.5: *Examples for cabin integration datums*

The *StructureComponents* (or cabin integration datums) can be, for instance, the floor level, the side shell area or the crown area, as figure 4.5 shows. Similar to the *CabinModules*, the 3D geometry data is not mapped on the *StructureComponents* directly. Instead, attachment brackets or substructure components like tie rods are considered *SubComponents* and are linked to the *StructureComponents*. The datum features of the *StructureComponents* are not modeled explicitly in the CDL models because they are detail aspects of the fuselage architecture. For cabin tolerance management it is more important *that* the integration datums exist rather than knowing their exact definition features.

The linkages between the *StructureComponents* and the fuselage section datum do not need to be modeled explicitly. The corresponding attachments between the frames, the stringers, the shell plates, the floor beams and the crown area substructures and their functional descriptions are very complex and are not explicitly needed for a cabin architecture model. Hence, both virtual linkage coordinates as well as the tolerances between the integration datums and the fuselage section datum are stored in the slots *P1x* through *P4z* (see fig. 4.2) in the *StructureComponent* instances. The virtual coordinates may depend on the mentioned architecture geometry parameters, so the equations for the coordinates can be defined on the class level within a scenario-specific class diagram.

4.2.2 Mechanical Interfaces and Installation Steps

Aside the concept of the *PhysicalComponent*, the second key aspect with manifold multi-domain correlations are the *MechanicalInterfaces*. The discussion in chapter 1 about the role of interfaces for architectures as well as in chapter 2 about the role of tolerancing indicate that the existing interface concepts [70, 104, 106, 170] can be extended to describe interfaces not only for assembly analysis, but for more general interface modeling purposes within the context of this thesis [93]. Consequently, a clustering of interfaces for mechanical integration aspects according to their *interface function* is proposed:

- interfaces fulfilling a *spatial distance function*
- interfaces fulfilling a *locating function*
- interfaces fulfilling a *fixing function*

The *MechanicalInterfaces* are hereby considered abstract functional objects with different interpretations for different aspects, depending on the mapped function. Consequently, a mechanical attachment interface can be either a *kinematic linkage* (locating function) or a *tolerance target* called *PKC* (spatial distance function) for tolerance analysis [27, 28, 93] or a *load interface* (fixing function) – or even combinations of the three[79], as can be seen in the CDL class diagram (see fig. 4.2).

Each *MechanicalInterface* instance can be interpreted as an *installation step* and therefore as an implicit design constraint for the installation sequence in FAL[80]. For this purpose, it inherits from the class *InstallationStep*. In this context, an *InstallationStep* is considered a single working step during the installation process of a *PhysicalComponent* with the corresponding sequence number (slot *ISNumber*) and working time and cost (slots *ISTime* and *ISCost*). The latter ones can for instance be synthesized with empirical estimation formulas (see fig. 4.3), or static values are applied. Additionally, the *MechanicalInterface* objects have slots for special naming and numbering conventions. Each *MechanicalInterface* instance is likened to two *FunctionalGeometryFeatures* belonging to the interfacing *PhysicalComponents*.

From an architectural point of view, the overall number of interfaces within a technical cabin integration scenario can be used as architecture analysis parameters (AAPs) to measure design complexity, as will be described in section 4.3.

4.2.2.1 PKC (Performance Key Characteristics)

A *PKC* (performance key characteristic) is an interface of an assembly located at gaps or a split line. The two interfacing components are not in contact, but their geometrical shapes are bound to a function. Thus, it serves a *spatial distance function*[81]. The function can either be appearance-related (e.g., a split line between two cabin modules), operations-related (gap for a flap mechanism, e.g., of a overhead stowage bin door) or installation-related (e.g., pin-hole coincidence). Speaking with tolerancing language, a *PKC* needs to have a nominal value and a tolerance specification according to ISO 286-1:2010.

On top, it can reflect a dynamic in-flight deflection limitation requirement between two *FunctionalGeometryFeature* objects. To ensure these functions, the tolerance (slot *PKCtoleranceRequirement*) and a minimal gap width (*PKCminimalGapRequirement*) are specified. The type of the *PKC* is saved in the attribute *PKCfunction*. There is an attribute for the nominal size within the slot *PKCnominalValue*.

[79] The mentioned supporting features [138] and the contact joints [104] usually are *only LoadInterfaces* without being *KinematicLinkages* in parallel. Conventional mate joints [104] consequently are *MechanicalInterfaces* which fulfill both the locating function *and* the fixation function, and which are implicitly constrained by the design. To complete the listing, *MechanicalInterfaces* which are *only KinematicLinkages* are mates that are free to be chosen for the manufacturing process [170]. However, this means that they require adjustment or tooling and thus imply type-II assembly processes.

[80] This represents a comparable concept to the 'assembly joints' [170].

[81] *PKC* does not inherit from *KeyCharacteristic* since it is a mechanical interface and not a feature tolerance. Here, common terminology is not consistent with the ontology, leading to pragmatic class diagram definitions (refer to footnote 74, page 55).

4.2.2.2 Kinematic Linkages

A *KinematicLinkage* is an abstract point in space where the position of two *PhysicalComponents* is determined relative to each other. This is accomplished by blocking one ore more kinematic DOF in between. This represents the aforementioned *locating function*. The functional directions of the blockings can be described explicitly within the slots called *FunctionalDirection1* and *FunctionalDirection2*[82]. The special type of the *KinematicLinkage* describes which combination of DOF is blocked (see fig. 4.6):

- A *2wayLocator* blocks one translational DOF.
- A *4wayLocator* blocks two orthogonal translational DOF, which usually contribute to the components's secondary and tertiary datum plane.
- A *Rotated4wayLocator* serves the same function as the 4wayLocator, but one functional direction belongs to the primary datum plane.
- Locally, a *LocatorPlaneElement* behaves like a *2wayLocator*, but three or more *LocatorPlaneElements* act together and lock one translational and two coupled rotatory DOF.

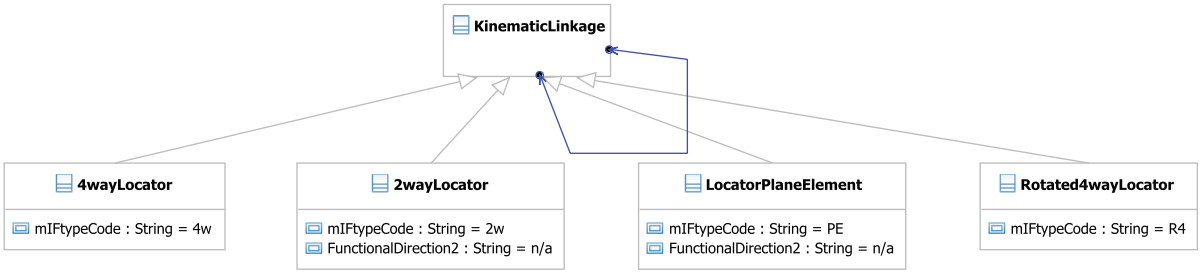

Figure 4.6: *The KinematicLinkage classes required for the CDL class diagram*

As already mentioned in chapter 2, most cabin lining panels have a linkage system that uses the same geometry features as their corresponding component datum system[83]. The lining panels which are modeled within this work use a *3-2-1 linkage system* (fig. 4.7) and a *3-2-1 datum system* respectively. Within the CDL implementation, the *3-2-1 linkage system* consists of three or more *LocatorPlaneElements*, which together block the first translational and the first two rotatory DOF of the panel installed in the fuselage. If more than three *LocatorPlaneElements* are modeled (which is usually the case to prevent edges from hanging down and from fluttering), this results in an over-constrained system in the primary direction, for which however simple calculation workarounds are possible within the convenient CAT software. Additionally, a combination of one *4wayLocator* and one *2wayLocator* for the remaining three DOF is needed. Both together block the remaining rotatory DOF and one translational DOF. The *4wayLocator* also blocks the last translational DOF.

Further special linkage systems are the *lateral stowage bin linkage system* and the *center stowage bin linkage system* (fig. 4.8). The *lateral stowage bin linkage systems* consists of two *Rotated4wayLocators* and two *LocatorPlaneElements* for the primary linkage plane (one translational DOF, two rotatory DOF)

[82] See fig. B.18 in Appendix B, page 146, for the inheritance relationship.

[83] A *datum system* describes the way a component is fixed on a measurement machine to measure its tolerances (e.g., see fig. 2.3). The *linkage system* describes how a component is installed within an assembly concerning its kinematic DOF. For most cabin components the *datum system* and the *linkage system* use the same geometry features since engineering heuristics and experience show that this usually minimizes the assembly tolerances. The comparison of figures 4.7 and 4.9 shows this principle. See sections 2.1 and 2.3 for more detailed explanations.

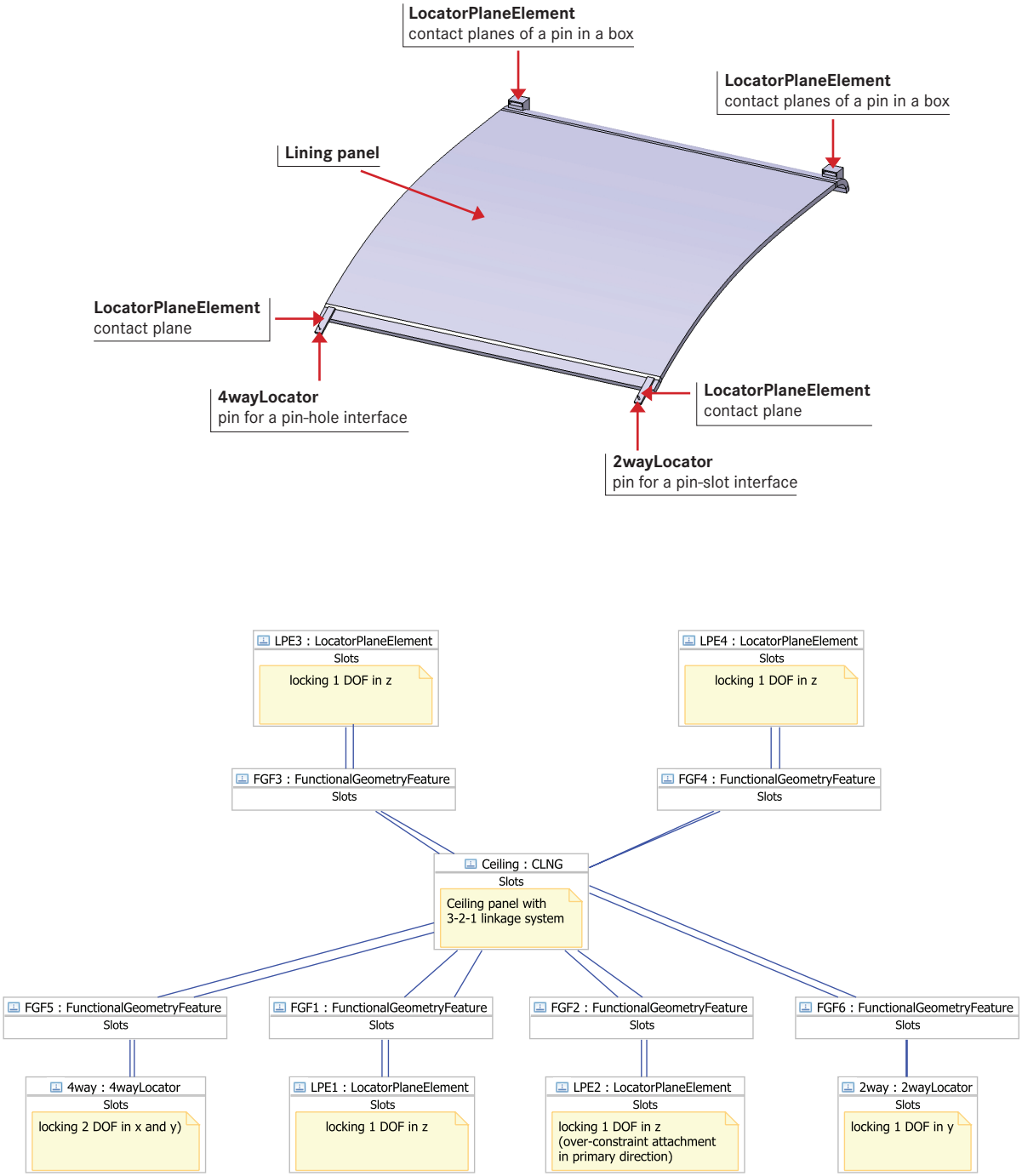

Figure 4.7: *3-2-1 linkage system of a cabin furnishing ceiling panel (top) and the abstraction using CDL vocabulary in UML (bottom). The class CLNG represents a ceiling panel as particular cabin module and inherits from the class CabinModule (see Appendix C).*

and for the secondary linkage axis (one translational and one rotatory DOF), as well as of one *2wayLocator* to block the remaining translational DOF of the module. The *center stowage bin linkage system* comprises four linked *Rotated4wayLocators* for the primary linkage plane and for the secondary linkage axis (middle axis), together with one *2wayLocator*. For all of these linkage systems, an individual setup is required concerning the position and orientation of the *FunctionalGeometryFeatures* and the linked *MechanicalInterfaces*) for each individual cabin module.

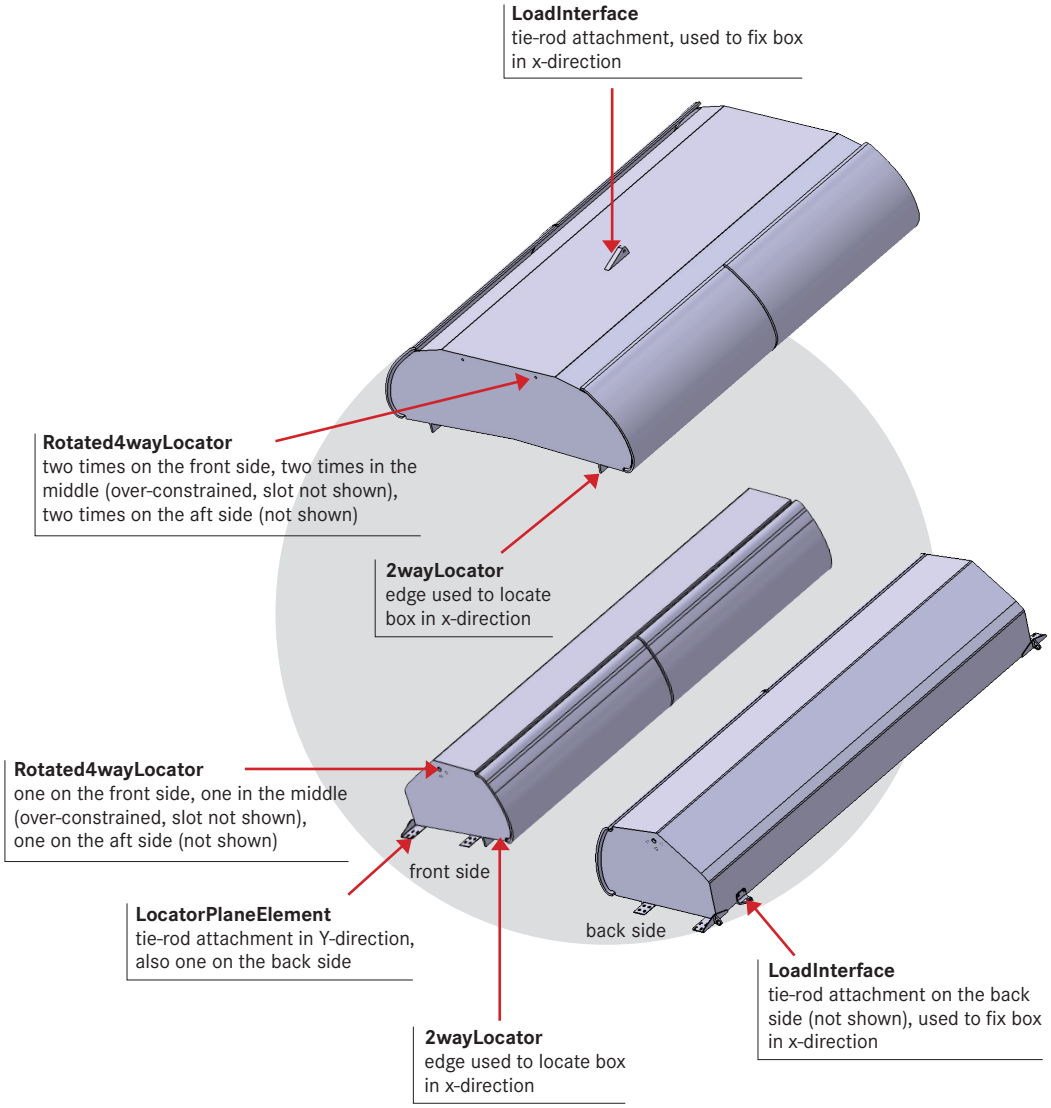

Figure 4.8: *Special linkage systems for stowage bins*

4.2.2.3 Load Interfaces

A *LoadInterface* is an abstract point in space where two *PhysicalComponents* are interconnected so that loads and momentums can be transferred between them (*fixing function*). Comparable to the *Kinematic-Linkage*, the type and the direction of the transmissible linear forces and the momentums need to be described explicitly within the slots called *FunctionalDirection1* and *FunctionalDirection2*.

For CDL models, the modeling of *LoadInterfaces* is primarily needed for architecture evaluations using architecture analysis parameters (AAPs, see section 4.3). But – if required – they could be used as boundary conditions for FEM models. Within the context of the CDL, there are only static *PhysicalComponents* and no movable parts, such as for example flaps, rudders or landing gear kinematics. Nevertheless, dynamic in-flight deflections can occur, and the load interfaces play an important role here for potential FEM based load and deformation analysis.

If a *MechanicalInterface* is classified as *LoadInterface* and as *KinematicLinkage* at the same time, it locally fulfills both the fixing and the locating function. This superposition is required for all mechanical interfaces of a physical component in order to get type-I assembly processes without any adjustability during installation. This is the case for the cabin compartment modules following the *3-2-1 linkage system* (e.g., see fig. 4.7). The x-location[84] and the x-fixation of the lateral and center stowage bins modules are situated at two different locations using two different *MechanicalInterface* objects. Thus, the installation of these modules has to be classified as type-II assembly.

4.2.3 Functional Geometry Features and Datum Features

The functions of *MechanicalInterface* objects take effect at the so-called *FunctionalGeometryFeature* objects which are special geometry features of a *PhysicalComponent*[85]. Figure 4.9 shows the detailed 3D geometry and the the abstraction of *FunctionalGeometryFeatures* with a focus on the physical integration aspects. As can be seen in figure 4.7, *FunctionalGeometryFeatures* are linked to *MechanicalInterfaces*.

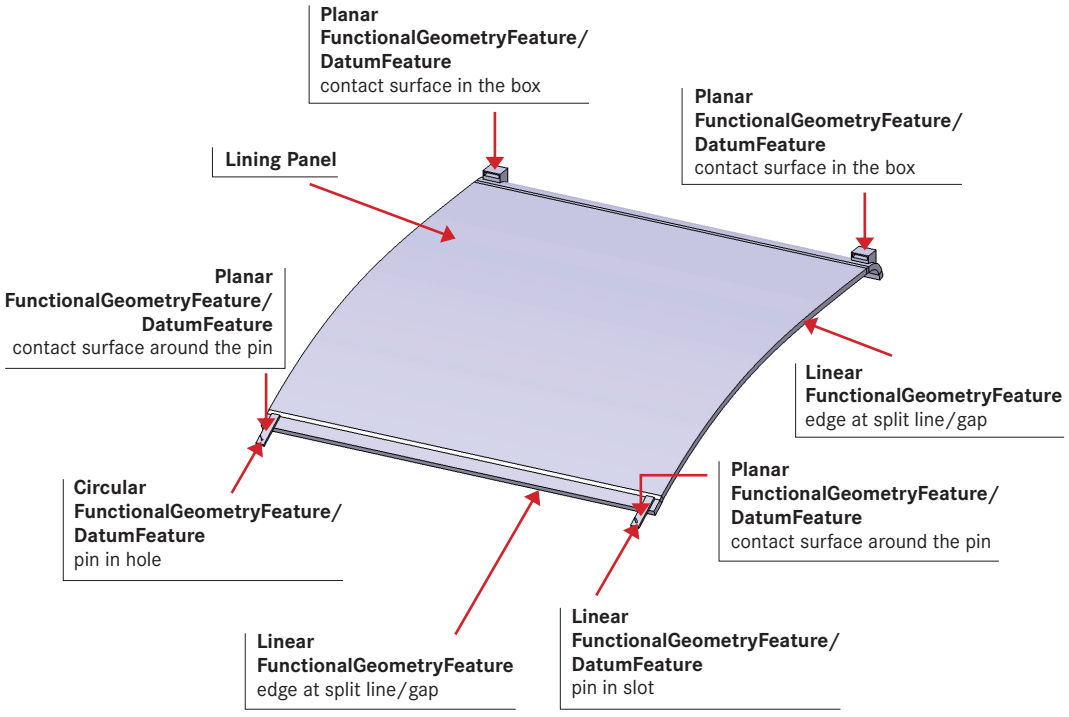

Figure 4.9: *FunctionalGeometryFeatures of a ceiling panel acting either as datum features or having MKC tolerances. See fig. 4.7 for the corresponding KinematicLinkages.*

[84] As convention, the x-direction of an aircraft's CAD coordinate system (see footnote 30, page 23) goes backwards along the longitudinal axis, the y-direction goes transverse, while the z-direction goes upwards.

[85] E.g., WHITNEY talks of 'assembly features' [170] for assembly functions.

The level of implementation of the CDL foresees a simplification of the features to a single point. Despite this initial definition, it is possible to extend this simplification with special feature definitions representing, for example, planes, lines or circles with the correspondingly needed geometry parameters in a potential data library for *FunctionalGeometryFeatures*. The positional and directional attributes of the *FunctionalGeometryFeatures* can be subject to tolerance by linked *KeyCharacteristic* objects or can be declared as *DatumFeature* of the corresponding *PhysicalComponent* respectively *DatumSystem*.

DatumFeatures are special *FunctionalGeometryFeatures* representing a datum feature object according to ISO 5459:2011. Using these objects, the explicit datum system of a *PhysicalComponent* can be implemented into the CDL model. For conventional lining panels following a type-I assembly, the attachment features often are used as datum features. The comparison of figures 4.7 and 4.9 shows this principle. For type-II assemblies like overhead stowage bins, other features need to be used, such as for instance surfaces. Those *FunctionalGeometryFeatures* which are not classified as *DatumFeature* usually get the classifier *KeyCharacteristic*.

4.2.4 Key Characteristics and Tolerances

CDL objects of the type *KeyCharacteristic* have to be interpreted as geometric tolerances representing key characteristics (KCs) according to EN 9100:2010 and EN 9103:2005. They inherit from the class *Tolerance*, which is used in the sense of ISO 286-1:2010, and therefore has attributes for the tolerance range, the mean shift and for the distribution curve (*PlusMinus*, *MeanShift*, *DistributionType*). In order to separate between cabin and structure-related tolerances, the manufacturing tolerances of cabin modules are called *MKCs* (manufacturing key characteristics), while *AKCs* (assembly key characteristics) describe the position and orientation tolerances of *FunctionalGeometryFeatures* on the aircraft side. The type of the *KeyCharacteristic* along with an unique identification number is saved in the slot *KCname*.

The *KeyCharacteristic* objects are mapped to a datum, since any tolerance has to be seen relative to its datum. Those *FunctionalGeometryFeatures*, which are not specified as *DatumFeature* can have associations to *KeyCharacteristics*. This way, the GPS tolerance of a geometry feature is specified relative to a datum using UML objects. In order to model group tolerances, the classifier *ToleranceRefinement* can be chosen additionally, providing additional attributes for the tolerance range, mean shift and distribution type of the group refinement, which are *PlusMinusRefinement*, *MeanShiftRefinement*, *DistributionTypeRefinement*.

4.3 Architecture Analysis Parameters (AAPs)

The implicit meaning of each CDL model object has been outlined in the previous section. The main purpose of CDL models, however, is to allow for a joint analysis of all model objects together, enabling the comparison and the evaluation of technical scenarios which consist of multiple objects. In this context, the formal analysis of cabin architecture models with CDL can be considered the measurement of design and architecture complexity using suitable metrics [43], here called *architecture analysis parameters* (AAP). In principle, this can comprise any computing of analysis parameters out of structured formal data, such as building the sum of values or counting the total of a certain kind of design objects. If comparison data is available, it is possible to set these measured system parameters in relation to other reference parameters.

A couple of such AAPs are proposed below. These parameters should neither be considered as ‘the only suitable’ parameter set, nor as a ‘complete’ description – especially not as independent and dimensionless similarity numbers being evaluation parameters in the sense of the pi-theorem [131, 133]. Nevertheless, these analysis parameters allow to describe physical integration aspects of cabin architectures in a pragmatic way.

4.3.1 AAPs related to Cabin Modules

Number of cabin compartment chain module types The more different cabin compartment chain module types exist, the more development and NRC efforts including efforts for detailed design, stress analysis, qualification and documentation have to be taken. Additionally, the integration complexity increases, meaning for instance more correlations between cabin modules such as PKCs. In contrast, fewer cabin compartment module types can mean that the modules get larger, which can be difficult for manufacturing including for the manufacturing tolerances and for the installation.

Total number of cabin compartment modules The total number of cabin compartment modules indicates the DMU integration efforts. Without a consistent standardization of the mechanical and system interfaces, more modules also means more development efforts. As an alternative, the parameter can be expressed in relation to the overall length of the cabin or cabin segment, to get comparable values for different aircraft with different cabin compartment lengths.

Mass of cabin compartment modules / mass of attachment brackets / total mass / ratio Like for any other system or component of the aircraft, the mass of the cabin compartment modules, the mass of the corresponding attachment brackets and their total mass are key aspects of cabin architectures. Certainly, these mass values need to be considered and evaluated together with the corresponding aircraft structure mass values, as only both together can be used to evaluate the mechanical architecture efficiency with regard to the overall aircraft mass performance[86].

The ratio of both values expresses how well cabin modules and attachment brackets are physically decoupled [157]. A low ratio indicated an architecture that needs a large substructure mass to integrate the cabin modules. This can go back for instance to requirements like self-locking brackets, which increase mass due to additional functional requirements for the brackets. A high ratio indicates a weight-optimized bracket application, which however may impact cabin module manufacturing costs or installation time and cost.

Overhead stowage bin loading volume The total overhead stowage bin loading volume is one indicator for the cabin compartment performance[87]. Airlines favor large overhead stowage space, as the passengers often prefer to have their hand luggage within reach. The smaller the stowage bin compartments are, for instance, due to short fuselage frame distances or due to a stowage compartment module architecture with short modules, the more unusable space in the gaps and split lines is wasted.

4.3.2 AAPs related to Mechanical Interfaces

Number of generic PKC types This AAP indicates in general the quantity of different PKCs, which lead to both tolerance management efforts during the development phase (non-recurring engineering cost) as well as to individual checking and measurement processes in the FAL (recurring manufacturing cost).

Total number of PKC applications This AAP summarizes the effects of generic PKC diversity and of the cabin (segment) length and quantifies the explicit FAL checking and measurement steps. The AAP can be set in relation to the cabin (segment) length in order to make length-independent conclusions.

[86] For instance, if cabin modules contributed to the stiffness of the fuselage, they would have an increasing mass. Conventionally, this is regarded as a poorer design than cabin modules without this function. On the other hand, this could lead to reduced loads impacting the primary structure and thus also to potentially reduced fuselage mass. Consequently, both cabin *and* fuselage mass should only be evaluated jointly concerning aircraft mass performance.

[87] See chapter 6 for a discussion of this AAP.

Total number of optical PKC / Total number of installation PKC / ratio The first parameter indicates how many optical PKCs there are in the cabin segment. It can be considered an analysis criteria for how good the industrial design 'forgives' tolerances: the more optical PKCs there are, the more gaps and transitions are visible and not hidden. The total number of installation PKC is a measurement parameter for the number of tolerance compensation features such as slots and oblong holes, which need to be toleranced in order to ensure installability. It will be high for type-I installations, and will be lower for type-II installations. These parameters can also be checked on the cabin module level to compare different module concepts directly.

The ratio of both can tell how the cabin architecture is driven by industrialization requirements (installation PKC) in relation to customer requirements (appearance PKC). An architecture with many type-I cabin modules and 'tolerance-forgiving design' will have a larger number than an architecture with several type-II cabin modules with many visible tolerances: higher numbers are better for FAL processes, but need a tolerance-proof design, whereas lower values are worse for FAL processes, but may have less restrictive tolerance requirements.

Number of interface types with mechanical functions This AAP is an indicator for the overall design complexity and in particular for the generic development efforts. It depends on the number of cabin compartment chain module types, but additionally expresses the efforts for mechanical integration. More interfaces may lead to more non-recurring development costs, but also to increased recurring manufacturing costs in the FAL.

Total number of kinematic linkages The parameter indicates the complexity of the mechanical interface architecture between the cabin compartment modules and the aircraft structure concerning DMU integration efforts including customization efforts. For a length-independent comparison of different cabin architectures, the ratio per cabin (segment) length can be used.

Number of cabin-to-cabin interfaces / number of cabin-to-structure interfaces / ratios These AAPs aggregate the number of cabin-to-structure and cabin-to-cabin interfaces as the sub-sum of the total number of interfaces. Cabin-to-cabin interfaces are functional couplings between cabin modules in the design graph and can lead to tolerance restrictions and additional load paths. Cabin-to-structure interfaces are attachments between cabin modules and the fuselage. The smaller the ratio between these two numbers, the more cabin-to-cabin interrelations there are, leading to an increasing design complexity.

Total number of 'only fixing' interfaces / total number of 'only locating' interfaces / ratio to total number of mechanical interfaces These AAPs indicate how many mechanical interfaces only serve the fixing function or only serve the locating function respectively. If type-I installation processes are aimed at, which are free of adjustment steps and free of installation loads (e.g., push the part into the final position), 'locating-only' interfaces must be avoided, since they require an additional step for the implementation of the fixing function. The example of the stowage bins (fig. 4.8) shows this situation.

If the ratio of the locating-only interfaces in reference to the total number of mechanical interfaces is 0%, then all cabin compartment modules are installation type-I. Formally, the number of interfaces is then minimized, due to a maximal coupling of the mechanical functions. However, a very good cooperation of tolerance management, stress and deflections is required to enable type-I installations in FAL for all cabin modules. If the ratio is larger than 0%, the architecture contains some type-II installation processes for cabin modules. This reduces the functional coupling between stress, tolerances and deflections and may simplify some complex technical scenarios.

4.3.3 AAPs related to Tolerances

Total number of different integration datums If a fuselage comprises too few integration datums for cabin integration (e.g., only the floor level as z-reference), this increases the functional tolerances: long distances to the datum features mean that the positional tolerances of the structure interface points get larger along with their contribution in the 3D tolerance stack. On one hand, more integration datums reduce the distance between toleranced feature and datum, which can deflate this phenomenon. On the other hand, the efforts for the fuselage architecture as well as the design and manufacturing complexity increases. Additionally, those PKCs which are between modules linked to different integration datums are impacted by datum shift tolerances.

Number of different AKC types Each AKC type represents a generic manufacturing constraint due to a functional tolerance requirement (PKC). The more different AKC types there are, the more complex the definition of the structure component manufacturing processes.

Total number of AKC applications Each AKC represents an individual manufacturing constraint and needs to be controlled during manufacturing, which is associated with impacts on manufacturing time, cost and quality. The more AKCs, the more complex is the definition of the manufacturing processes. Additionally, every AKC leads to recurring manufacturing cost. The ratio per cabin (segment) length indicates the number of length-independent manufacturing constraints.

Total number of AKC applications with a link to an appearance PKC / ratio to total number of AKC applications Unlike MKCs, the number of AKCs does not necessarily decrease if the number of appearance-related PKCs is reduced, as AKCs are often required for installation PKCs only. The larger the ratio, the higher the influence of appearance requirements is on the aircraft structure design. The smaller the ratio, the higher the influence of cabin installation-related requirements is on fuselage and secondary structure design.

Total number of AKC cabin-to-cabin / total number of AKC cabin-to-structure / ratios Cabin-to-cabin AKCs are restrictions for FAL installation processes. The more interfaces there are between cabin modules, the more complex the requirement interdependency in the design graph is and thus the more complex the design is. This means that when the ratio is smaller, less cabin-to-cabin interactions lead to an increasing design complexity: the larger this number, the larger the design complexity of the design graph representing the cabin architecture scenario.

Number of AKC cabin-to-structure with refinements This sub-sum of the overall AKCs indicates how many tolerance requirements are locally coupled and therefore add constraints and complexity to the manufacturing processes. If too many tolerance refinements need to be specified, the integration datums probably have not been defined well looking at the functional needs[88].

Ratio of total number of AKC in relation to total number of interfaces with mechanical function The higher the ratio, the more complex the design is. In particular, a high ratio can indicate that there is a highly integrated design with the intention to reduce the number of mechanical interfaces. For instance, this can be a weight-optimized and very modularized design, but the manufacturing and installation processes can be more difficult than for a design with a smaller ratio.

[88] Large tolerance groups can be considered as local integration datums for functional purposes alternatively. For large groups it may make sense defining a real additional integration datum rather than tightening manufacturing tolerances with over-constraint group tolerances.

Number of different MKC types Each MKC type represents a generic manufacturing constraint due to a functional requirement (PKC). The more MKC types there are, the more complex the definition of the cabin module manufacturing processes becomes. 'Tolerance-forgiving' design in this case means a low number of different MKC types.

Total number of MKC applications The more MKCs, the more final measurement efforts need to be taken, influencing time, cost and quality of the cabin module manufacturer. The ratio per cabin (segment) length indicates the number of length-independent manufacturing constraints for the cabin modules.

Total number of MKC applications with a link to an appearance PKC / ratio to total number of MKC applications If type-I installation processes are dominant, MKCs are mostly needed for appearance PKCs. Values smaller than 100% indicate, that the cabin modules are designed to have type-II installation processes in FAL. Consequently, some MKC tolerances need to be controlled during manufacturing just to ensure installability. The corresponding cabin architecture is more complex than one without MKC tolerances for installation purposes. Similar to the AKCs, the ratio to the total number of MKC applications indicates, how many tolerancing efforts are needed.

4.3.4 AAPs related to Installation Aspects and Costs

FAL Mh estimation bottom-up An important AAP concerning the FAL installation processes is the estimation for the overall working time needed to install the cabin into the aircraft. Modern cabin architectures aim at limited installation time. However, this time needs to be considered and evaluated in conjunction with the installation time and cost for the fuselage at MCA.

Installation Mh / work preparation Mh / ratio of both This is on one hand the sum of all installation-related working time in FAL, on the other the sum of all preparation-related working time in FAL (carry modules into the aircraft, fetch tools and jigs etc.). The corresponding ratio reflects, whether there is high preparation work, for instance due to many tool-based installation processes. The higher the ratio, the more optimized the architecture concerning smart installation processes is. However, if the ratio is very high, the design and manufacturing repercussions for MCA - for instance strict tolerance or process requirements - need to be considered.

Total number of FAL installation steps This parameter counts all individual steps to install all cabin compartment modules. Therefore, each mechanical interface is considered as one installation step. To eliminate the influence of the length, the parameter can also be set in relation to the overall length of the cabin or cabin segment.

Rate of cabin compartment chain modules with type-I installation The more type-I installations designed, the fewer the expected installation efforts in the FAL, but the tolerances (MKC and AKC) need to be stricter in order to enable these installation processes. Alternatively, the PKCs for optical aspects could be increased with the corresponding impact on the appearance quality.

Cabin compartment manufacturing cost / module and bracket purchasing cost / installation cost As proposed before, the cabin compartment-related manufacturing cost can be broken down into the modules' and brackets' purchasing cost and into the FAL installation cost as primary recurring cost indicators. As an analysis, the corresponding cost budgets can be summarized.

4.4 Generic CDL Modeling and Analysis Process

The following methodological framework explains how models using the CDL vocabulary can be built up in a generic way and how technical scenarios can be analyzed using the CDL models. Since the tolerancing-related model aspect are the largest contributors to the final model space, it is helpful to choose the rule sequence in dependence on the tolerancing process as presented within section 2.6.

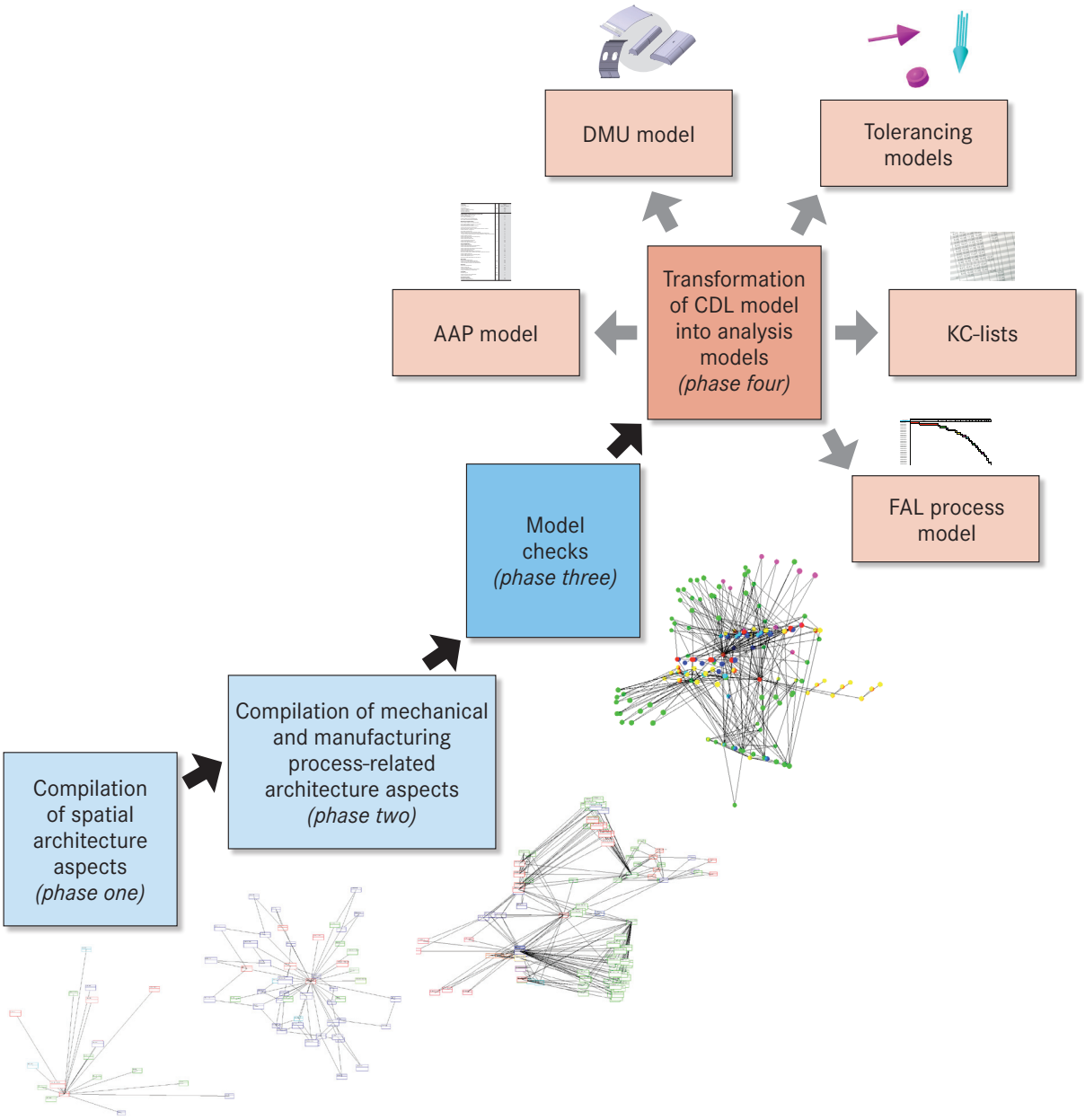

Figure 4.10: *Generic CDL modeling and analysis working process (compare with fig. 1.7, page 13)*

To start it is necessary to create a case-specific class diagram inheriting from the generic CDL class diagram. The class diagram should be enriched by CAD data in the required granularity linked to the corresponding classes. As described by figure 4.10, the following synthetic compilation of the analysis

models consists of four phases. The first and second phase are the actual modeling phase – the phases, in which the design graph consisting of CDL vocabulary grows rule-based using model-to-model transformations (first two steps in fig. 4.10).

Within the *first phase*, the technical design and architecture framework needs to be set up. This comprises the compilation of spatial architecture aspects like the definition of architecture input parameters (e.g., definition of the frame length or of comparable geometry patterns), a conceptual arrangement of the *CabinModule* objects within the virtual fuselage, the definition of the structure integration datums (*StructureComponents*) and the definition of gaps (*PKCs*) with product performance relevance including tolerance requirements and deflection limitations.

The *second phase* is the detailing of the model, which comprises the compilation of mechanical and manufacturing process-related architecture aspects. In particular, this means the application of *SubComponents* and geometry data to increase the level of granularity of cabin modules, structure integration datums and brackets to the required depth, the application of *MechanicalInterfaces* with the needed mechanical functions and the corresponding *FunctionalGeometryFeatures*. Additionally, the model can be enriched with further domain data (mass, cost, manufacturing process information etc.).

Ensuring syntactic, semantic and pragmatic correctness of the data models constitutes a key aspect of design languages in use [135]. *Syntactic correctness* needs to be ensured by the compilation algorithms during the rule execution. Only rules which are syntactically correct should to be executable [135]. *Semantic correctness*, however, depends on the individual data models and on the corresponding class models. Therefore *phase three* for model checks is used after the design graph has reached its final size and before the analysis transformations are conducted. For instance, it is necessary to check whether all required class definitions are set, whether all required instances and slots exist and whether all slot values have the correct data type and format. Additionally, manual *pragmatical model checks* should be performed by the user before going into the analysis phase.

The *fourth phase* starts when the design graph representing a cabin architecture scenario is finished and checked. The corresponding analysis transformations have to be a reasonable combination of model-to-model and model-to-text transformations, depending on the explicit software implementation. The analysis methods are constituted by the scope of modeling as proposed in section 4.1 and comprises the following analysis methods[89]:

- Calculation of the AAPs
- Compilation of DMU data
- Tolerance analysis models
- Compilation of KC-lists for bi-directional exchange
- Compilation of FAL process overview charts

After the initial model analysis as described above is finished, further analysis methods outside the initial scope of modeling can be consulted. Considering methods like FEM-based load and deflection analysis, implementing links to functional systems analysis models, more detailed cost prediction or manufacturing planning methods or ergonomics analyses all depend on design- and scenario-specific technical or functional circumstances. If needed, the CDL data models therefore could be transformed into further data models, as soon as the correlations between the CDL class model and the new target class model are described.

[89] These analyses can be considered as part of the checks for *pragmatic model correctness* [135].

Chapter 5

Implementation

5.1 Software Interfaces

For this thesis, the representation of graph-based design languages in UML format has been chosen for the software-based implementation example of the CDL model framework [15, 128]. To fulfill the intended analysis task of mechanical cabin architectures including tolerancing, it is necessary to deploy some new software interfaces called *plugins* for the required information processing[90] [75]:

- Plugin constituting a software link to CAT software following a kinematic modeling approach
- Plugin to establish the scenario-independent CDL class model including some general routines
- Plugins to transform the CDL models to analysis and visualization models

Figure 5.1 introduces the new plugins that have evolved accordingly and indicates their hierarchical dependency in the software environment. The following subsections will provide an overview about these new software interfaces.

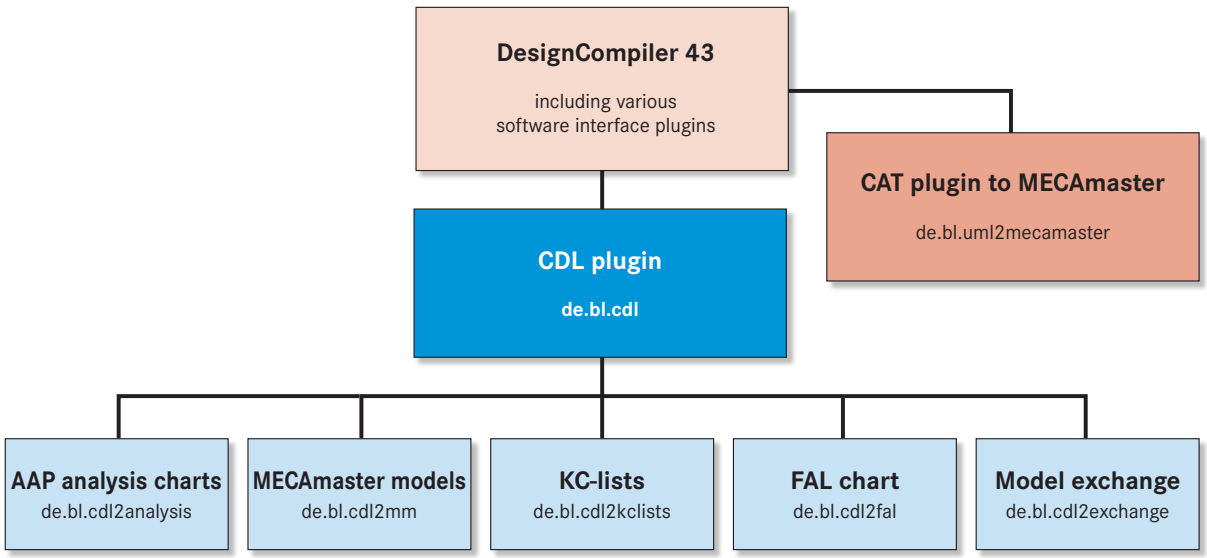

Figure 5.1: *Software interfaces for the software DesignCompiler 43 [75] to enable CDL modeling including tolerance analysis*

5.1.1 Interface for MECAmaster Models

The plugin *uml2mecamaster* is a new process chain for the aforementioned CAT software MECAmaster[91] and has been developed in the context of this thesis[92]. Unlike the following plugins with CDL-related functionalities, it can be used standalone with DesignCompiler 43 in oder to create m_m-files for MECAmaster calculations. It consists of two major packages called *uml2mecamaster.profile* and *uml2mecamaster.create_m_m*.

[90] E.g., see fig. 1.7, page 13.
[91] See subsection 2.3.2.
[92] The decision for the software interface to MECAmaster as CAT software is based on the fact that cabin tolerancing has to cope with simple kinematic interfaces which can be modeled in a fast and pragmatic way using MECAmaster. In addition to that, it is easy to create the m_m-files using model-to-text transformations, as will be shown in this subsection.

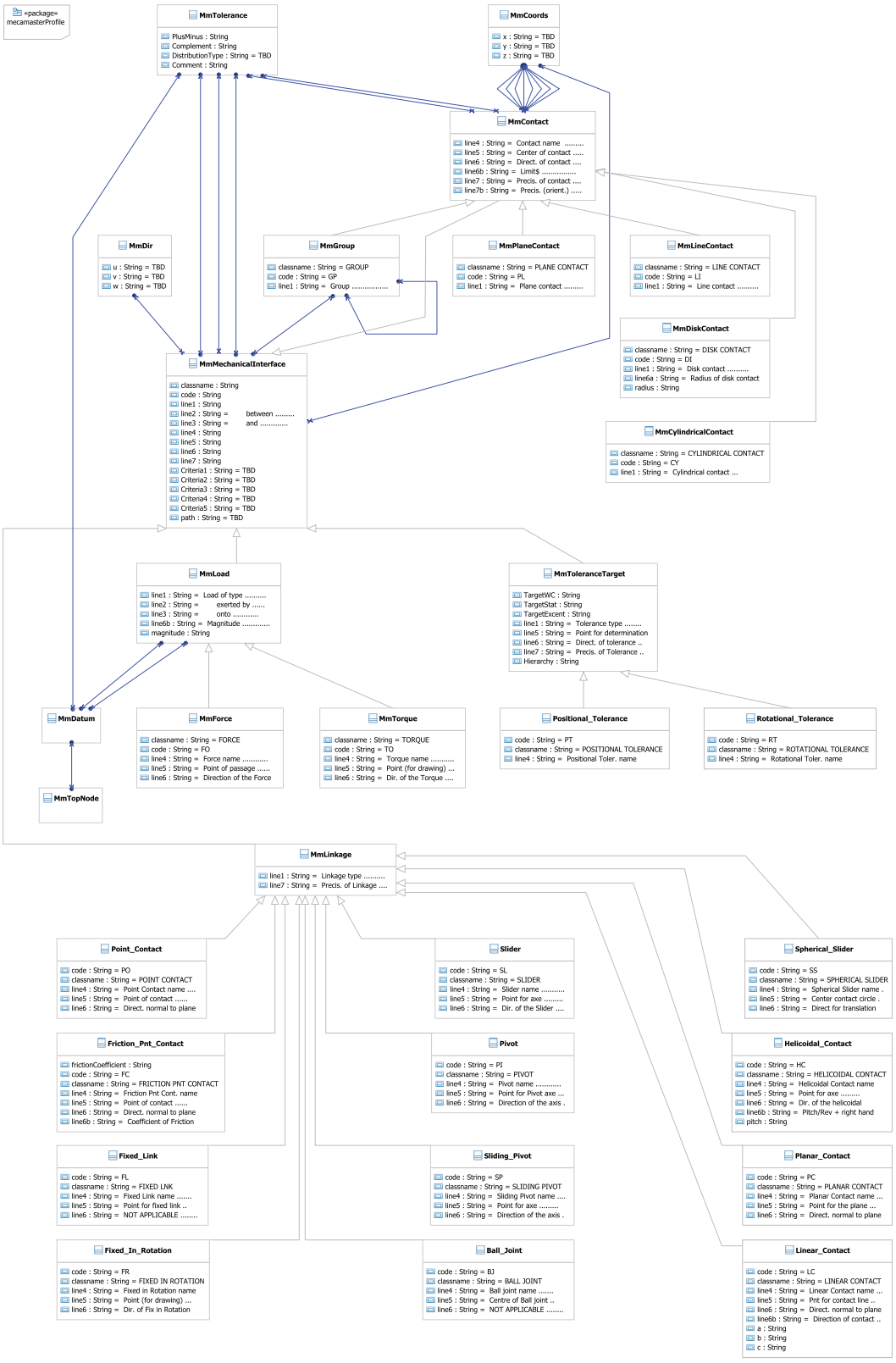

Figure 5.2: *Schematic overview of the MECAmaster class diagram in UML demonstrating the diversity of the class definitions needed to cover the full modeling range of MECAmaster [113]*

5.1.1.1 Packages and Routines

uml2mecamaster.profile The package *uml2mecamaster.profile* provides the definition of a UML profile for MECAmaster-specific classes (see fig. 5.2). In contrast to the DesignCompiler 43 interface to CATIA V5 used in other applications [16], no *UML stereotypes* are used in the MECAmaster UML profile, but MECAmaster-specific *UML classes* are defined, leading to a comprehensive class diagram (fig. 5.2). Any of the specific MECAmaster elements [113] mentioned in chapter 2.3.2 has a corresponding specific class in the UML profile, which inherits from the abstract class *MmMechanicalInterface*. Objects of this classifier can be regarded as empty container without any specific data. Specific data like the names of the interfacing datums (which are physical components), like coordinates, vectors and tolerances are stored in a modular way in separate instances linked to the *MmMechanicalInterface*. These classes are named *MmDatum*, *MmCoords*, *MmDir* and *MmTolerance* accordingly.

The modular object-oriented dependency allows flexibility concerning the data topology of each individual MECAmaster element (which is an instance of a class inheriting from *MmMechanicalInterface*) and the corresponding data block. The differentiation is supported by using the different associations between the objects to distinguish between different functions.

uml2mecamaster.create_m_m The package *uml2mecamaster.create_m_m* offers a set of model-to-text transformation routines. These routines transform a UML model consisting of MECAmaster UML instances into a valid MECAmaster input file (fig. 5.3). At first they generate the header of the m_m-file[93]. Then a loop transforms all *MmMechanicalInterface* instances and the associated data objects into corresponding data blocks. The data blocks can be created independent from others, which allows a sequential processing within the loop. The text output can be individualized for each *MmMechanicalInterface* instance. Interdependencies between the data blocks are consistent due to the modular data structure of the MECAmaster UML class model. During the transformation, syntax checks are performed to ensure syntactic correctness of the outcome.

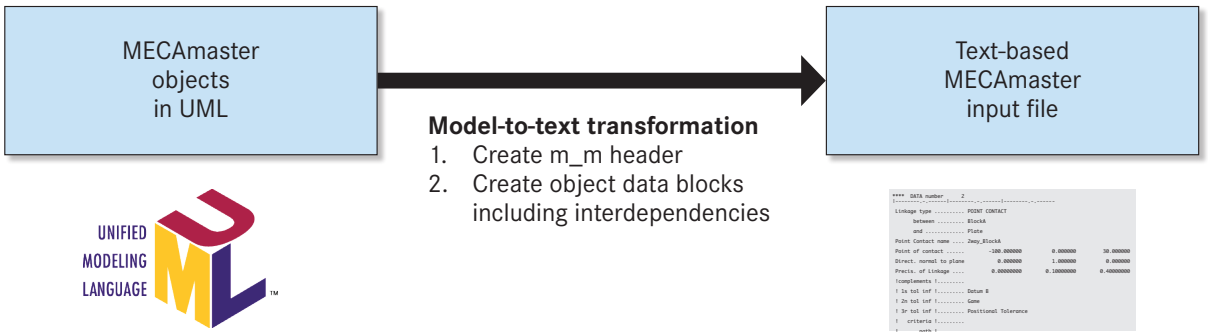

Figure 5.3: *Working principle of the plugin* uml2mecamaster

5.1.1.2 Creation of MECAmaster UML Models

The MECAmaster UML instances can be created in two different ways. First of all, the user can apply the MECAmaster profile directly in the context of design rules embedded in a design language. This way, the MECAmaster classes are used as design language, of which case-specific specific classes can inherit, or which can be used directly within rules. For instance, figures 1.8 through 1.10 in chapter 1 correspond to a MECAmaster design language model of the well-known example with the two blocks

[93] Subsection 2.3.2 shows a code example (fig. 2.11 on page 36).

and the plate. The rule as shown in figure 1.9 on page 15 is used to insert a *Positional_Tolerance* for the measurement of the offset T_O between the two blocks. Figure 5.4 below shows the rule 'A-2way'[94] which adds a *Point_Contact* linked to the instance 'BlockA'. This *Point_Contact* corresponds to the one shown in the GUI in figure 2.7 (page 32) as well as to the m_m-file code shown in figure 2.11 (page 36).

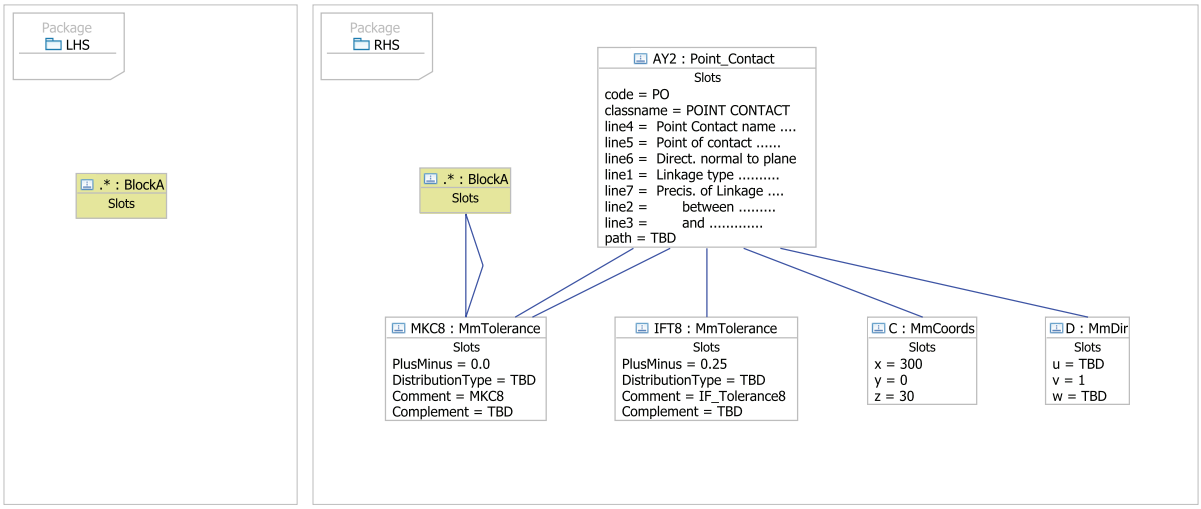

Figure 5.4: *Rule for inserting a 'point contact' in y-direction linked to block A*

Alternatively, the MECAmaster UML instances are created automatically by model transformations initiated by other plugins. For instance, the CDL implementation as described later works this way.

In any case, the plugin *uml2mecamaster* has to ensure that the resulting m_m-file is consistent and that its syntax is correct [135], so that it can be read in and proceeded by MECAmaster without errors. Therefore, multiple error handling routines are implemented directly during the generation of the data blocks. Among others things, the routines check if the slots of the MECAmaster UML instances have the correct data format and check if all required data objects are available by verifying that the required MECAmaster UML instances exist.

Using specific text formatting routines, the plugin ensures that the created text adheres to the sensitive m_m-file format rules. Any error is reported in an error log file, and the plugin only creates an m_m-file if no syntax error has been detected. This harsh error handling is especially useful if the plugin is used to transform automatically created MECAmaster UML instances. It has to be prevented that any mistakes or even systematic errors within such models propagate into the MECAmaster simulation. Due to the fast execution time of the plugin, it can be used as an error checking routine for automated MECAmaster UML model generation: if the error log file reports no errors and if the m_m-file is written out at the end, the automatically created MECAmaster UML instances are syntactically and semantically correct. If not, the originating model needs to be error-checked itself.

Of course, the plugin cannot ensure the pragmatic correctness [135] of the model – for instance, if the positions of the linkages are 'good' for the technical problem, or if the tolerance measurements are taken at the 'right' position. These tasks cannot be generalized and need to be handled by the generator of the UML model – be it an DesignCompiler 43 user or any other automatized software for MECAmaster UML model generation.

[94] See fig. 1.8 on page 14 for the corresponding rule sequence.

5.1.2 Interface for CDL

The core plugin of the CDL implementation is called *cdl* and consists of several packages:

cdl.profile Within the package *cdl.profile*, the plugin contains the UML implementation of the generic CDL class diagram as derived in chapter 4 as the major element (see fig. 4.2). The generic class attributes and associations are implemented accordingly.

cdl.operations According to the fact that not every model-to-model transformation can be easily expressed in graphical form, it is possible to write design rules using Java code [128]. This code is then either called by a graphical element in the activity diagram (*«javaRule»* [75, 136]) or can be executed manually after the complete execution of the design rules. Such operations are not specified directly in the UML class model, but are programmed in the separate but attached Java code. The CDL class operations are class-specific computation algorithms written in Java:

- For the mass budget of *CabinModule* instances, a Java routine checks for linked *Subcomponent* classes. If it finds one or more, their masses – which were calculated before by the implemented constraint processing technique of the design compiler – now are aggregated and saved to the slot *PCmass*. Comparable routines follow similar algorithms for the slots *PSCost*, *PSPreparationTime* and *PSInstallationTime*.
- The datum system of a *PhysicalComponent* can be modeled explicitly by applying the classifier *DatumFeature* to *FunctionalGeometryFeature* objects. In this case, a routine checks if the defined *DatumFeatures* lead to a valid datum system, where no unblocked DOF remain.
- For the *MechanicalInterface* objects, the position and orientation slots[95] are calculated using the corresponding position and orientation information of the associated *FunctionalGeometryFeatures*.
- Furthermore, it has to be checked if the explicitly modeled kinematic linkage system blocks all DOF in order to provide an explicit and non-ambiguous positioning of the component. Therefore, the functional direction slots of the KinematicLinkage objects are compared and it is determined whether the blocking directions are independent.
- The slot *KCname* of the *KeyCharacteristic* instances as well as the slot *IFname* of the *MechanicalInterface* instances are set according to special naming conventions.

cdl.validations These Java functions perform some model validations on UML model level, for instance to ensure that slots values match with the required data type, to report wrong defining features or empty association member ends, or to ensure that specific associations required between CDL instances are made.

cdl.errorLog This code is used to output the error and warning messages, which are created during the execution of *cdl.validations* and *cdl.operations*.

cdl.common The public Java functions within this package are library routines like getter and setter routines for CDL objects. The package is published and therefore can be called for instance from the *cdl2*x plugins.

[95] The classifier *MechanicalInterface* inherits the slots *dx*, *dy*, *dz*, *FunctionalDirection1*, *FunctionalDirection2* from the classifier *DirectedPosition*, as can be seen in appendix B.

Model Name	-	Baseline
Trade Number	-	1000
Execution Date / Time	-	2012-01-31 / 19:46:25
Frame distance	mm	530
Length of F2F cabin compartment segment	mm	7950
Length of F2F cabin compartment segment (frame bays)	-	15
F2F frame architecture 1	-	4,0
F2F frame architecture 2	-	2,0
Cabin Module and Structure Component Data		
Number of different Cabin Module types	-	X
Number of Cabin Modules	-	X
Total number of Cabin Modules per m	1/m	X,X
Number of different StructureTopNodes types	-	X
Total number of considered StructureTopNodes	-	X
Mechanical Interface Data		
Generic number of different mechanical interfaces	-	X
Generic number of different mechanical interfaces per m	1/m	X,X
Total number of Kinematic Linkages	-	X
thereof: number of cabin-to-structure interfaces	-	X
thereof: Number of cabin-to-cabin interfaces	-	X
Ratio of cabin-to-cabin interfaces in relation to cabin-to-structure interfaces	-	X,X
Number of Kinematic Linkages per m	1/m	X,X
Total number of Load Interfaces	-	X
Number of Load Interfaces only (no Kinematic Linkage)	-	X
Ratio of coupled mechanical interfaces (Kinematic Linkage and Load Interface) in relation to overall number of mechanical interfaces (Kinematic Linkage and Load Interface)	-	X,X
Number of different PKC types	-	X
Number of PKC applications (to be checked by FAL quality department)	-	X
thereof: number of optical PKC	-	X
thereof: number of Installation PKC	-	X
Ratio of installation PKC per optical PKC	-	X,X
Number of PKC applications per m	1/m	X,X
Number of PKC applications per PKC types	-	X,X
AKC and MKC Data		
Number of different AKC types	-	X
Number of AKC applications (to be checked by quality department)	-	X
thereof: number of AKC cabin-to-structure	-	X
thereof: number of AKC cabin-to-structure with refinements	-	X
thereof: number of AKC cabin-to-cabin	-	X
Number of AKC applications per m	1/m	X,X
Ratio of total number of AKC applications with PKC link	-	X,X
Ratio of total number of AKC in relation to total number of Mechanical Interfaces	-	X,X
Number of different MKC types	-	X
Number of MKC applications (to be checked by quality department)	-	X
Number of MKC applications per m	1/m	X,X
Ratio of total number of MKC applications with PKC link	-	X,X
Mass Data		
Mass estimation bottom-up total	kg	X,X
thereof: mass estimation bottom-up ATA25 F2F	kg	X,X
thereof: mass estimation bottom-up ATA53 brackets	kg	X,X
Ratio of mass ATA25 in relation to mass ATA53 brackets	-	X,X
FAL Data		
FAL Mh estimation bottom-up	Mh	X,X
thereof: installation Mh	Mh	X,X
thereof: work preparation Mh	Mh	X,X
Ratio of installation Mh in relation to overall FAL Mh	%	X
Average installation Mh per installation steps	-	X,X
Cost Data		
Manufacturing cost	EUR	X
thereof: purchasing cost for Cabin Modules	EUR	X
thereof: installation cost in FAL	EUR	X
Performance Data		
Stowage bin loading volume	l	X,X
Stowage bin loading volume per m	l/m	X,X

Figure 5.5: *Spreadsheet-based visualization chart of AAPs generated by the plugin* cdl2analysis

5.1.3 Interface for AAP Charts

As mentioned before, the calculation of the AAPs as proposed in section 4.3 can be very time-intensive and error-prone. Due to the fact that the calculation methods for each parameter can be described formally, it is simple to embed the methods into re-executable software, which creates repeatable and consistent results. This plugin calculates and outputs the AAP into spreadsheet charts, as figure 5.5 shows. For each AAP, the corresponding calculation method is embedded[96].

After the data is written into the spreadsheet chart, any preformatted design is overwritten. So the plugin automatically opens the spreadsheet and an internal spreadsheet macro reformats the overview table. For comparative trade studies, multiple overview tables for different scenarios can for instance be used together with MS Excel-based visualization charts. Examples for such charts will be shown in chapter 6, where representative technical results are discussed making use of this visualization plugin.

5.1.4 Interface for Transformations from CDL to MECAmaster Models

When repetitive linkages and linkage patterns need to be reproduced [134], an automatized generation of MECAmaster input files can be beneficial. This plugin aims at providing this support. It is no 'general MECAmaster input generator', but is specialized for the transformations required within the context of cabin architecture analyses.

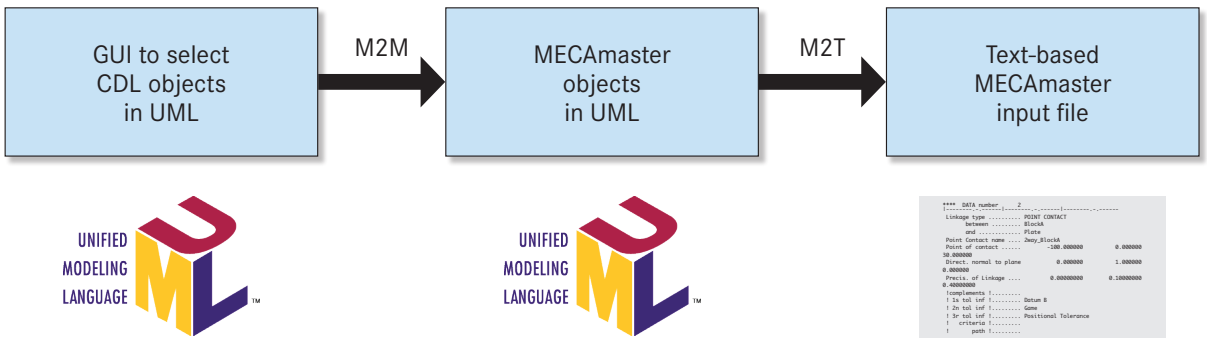

Figure 5.6: *Transformations between CDL in UML, MECAmaster in UML and text-based MECAmaster input files*

As a first step (see first box in fig. 5.6), a GUI offers to choose those *PKCs* from a list of available *PKC* instances which are to be calculated. Figure 5.7 shows the corresponding GUI whose code is contained in the package *cdl2mm.common*. After that, the routines from the package *cdl2mm.transformations* conduct a topological and parametrical search for certain linkage system patterns in the CDL model[97]. To do so, the plugin scans the CDL model for the *KinematicLinkage* instances constituting the linkage system of each individual *CabinModule*. If the linkage system matches one of the three patterns as described in figure 5.8, the plugin creates MECAmaster UML objects and compiles them to patterns corresponding to the CDL linkage system (fig. 5.6). The required attributes are either copied from the CDL UML instances to the MECAmaster UML instances, or are calculated and converted accordingly.

[96] Speaking formally, the information content of the model is not increased by these transformations. But for visualization, comparison and evaluation purposes it is easier to have *explicitly* visualized data, rather than sophisticated UML models using abstract UML diagrams [128].

[97] As mentioned in subsection 4.2.2, cabin compartment modules usually share three different linkage system patterns, which are a *3-2-1 linkage system* and special *lateral stowage bin* or *center stowage bin linkage systems*. The main difference between these systems is the number and the arrangement of the different elementary *KinematicLinkages* they use.

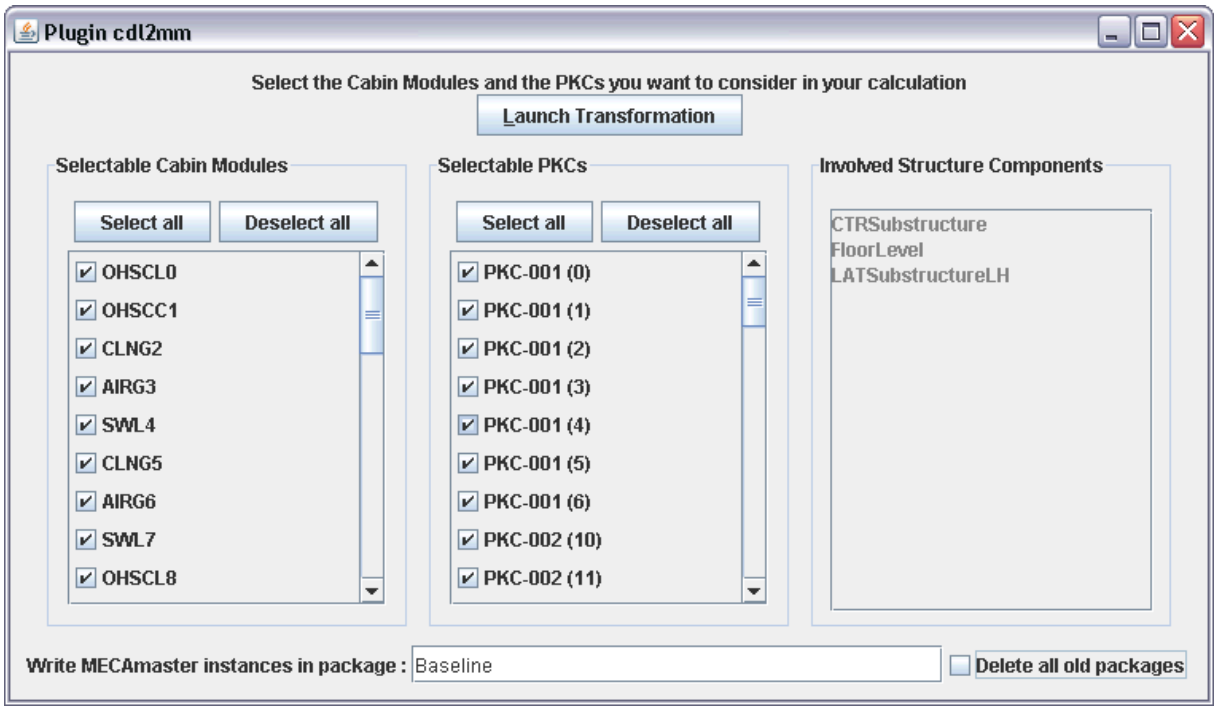

Figure 5.7: *GUI of the plugin* cdl2mm *to select instances for the data export to MECAmaster*

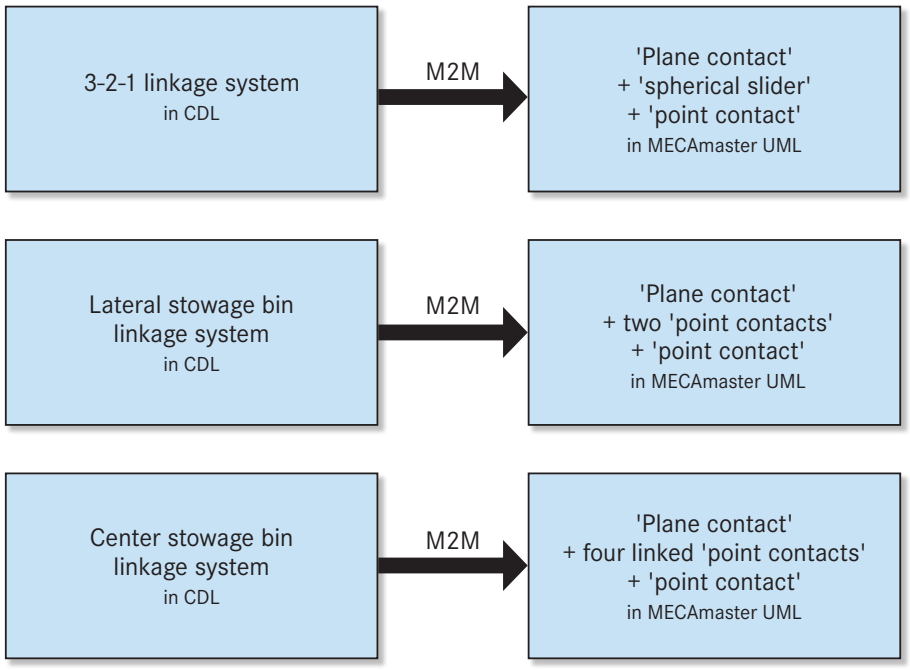

Figure 5.8: *Implemented topology transformations between CDL linkage systems and predefined MECAmaster linkage patterns*

As second step (see second box in fig. 5.6), the plugin loops over the *StructureComponents* in the CDL model and establishes MECAmaster linkages between the different *StructureComponents* acting as integration datum systems. Since the linkage systems between the *StructureComponents* are not modeled explicitly in the CDL models, the plugin assumes a *3-2-1 system* and uses the virtual interface points *P1x* through *P4z* and the tolerances in between as foreseen in the *StructureComponent* class[98].

In a third and last step (see last box in fig. 5.6), the MECAmaster UML instances are transformed into a MECAmaster input file by calling the transformation routines from the plugin *uml2mecamaster*. Of course, topological changes still can be made in MECAmaster in the sense of a standalone CAT environment. But after the results are obtained, the whole CDL model has to be re-executed using adapted input to reflect the investigated changes. A round trip[99] for *topological changes* at this point is neither recommended – due to the multi-domain repercussions, which can only be handled in the CDL model itself – nor is it simple to realize such complex bidirectional transformations of topologies. A round trip method for pure *tolerance value updates* is possible as presented in the subsequent subsection.

5.1.5 Interface for Tolerance Lists

The initial tolerance values come from estimation algorithms or from fix estimation values, which are implemented in use case-specific class diagrams directly in the CDL model. However, the initially specified tolerances often need to be adapted in a fast way during tolerance analyses. This should not happen inside the standalone tolerancing software, which is decoupled from the CDL model. Especially if one tolerance appears at multiple places, such manual updates can also be time-intensive and can lead to data inconsistency. Instead, manual changes should be fed back into the CDL model right away in order to enable a consistent reproduction of the tolerancing software input files and to ensure a later harmonization with the remaining CDL objects, which might face repercussions due to the changed tolerance values.

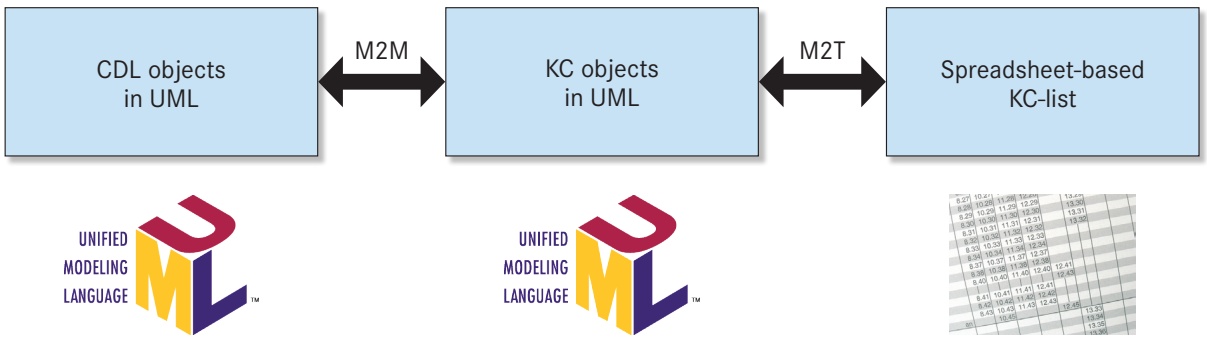

Figure 5.9: *Round trip engineering between CDL objects in UML, KC objects in UML and spreadsheet-based KC-lists*

Thus, a round trip process (fig. 5.9) is of interest at this specific point to cut back the execution time of a mono-disciplinary tolerance trade study and to ensure data consistency. Therefore, the routines within the package *cdl2kclist.transformations* collect all KC objects (*PKC*, *AKC* or *MKC* objects) in the CDL model and transform them into specific UML-based KC objects by model-to-model transformations. The corresponding UML profile for the KC-list classes is published under *cdl2kclist.profile*. The KC-list objects are then output into spreadsheets by model-to-text transformations. Additionally, the plugin

[98] See subsection 4.2.1 for further descriptions.

[99] 'Round trip engineering' or a 'round trip' process mean that software enables 'back and forth' model-to-model transformations between two data models.

offers the possibility to read back updated tolerance values from these spreadsheets into the KC-list objects and then into the CDL model. In this way, the spreadsheet-based KC-lists can be exchanged with neighboring engineering teams for updates and for analysis of repercussions.

This round trip engineering is possible for value updates between the spreadsheet, the KC-list objects and the CDL object with a well-defined bidirectional relation. It is not foreseen to append additional lines with new KC data into the spreadsheet and transform them into new CDL objects. Such topological changes should be managed within the CDL model at a higher level of abstraction or a different moment in the design compilation phase during the processing.

5.1.6 Interface for Installation Process Charts

The plugin *cdl2fal* (fig. 5.10) searches for all *ProcessStep* instances within a CDL model and writes them into a spreadsheet using model-to-text transformations using some sorting algorithms. Afterwards, a macro is executed within the spreadsheet file, which visualizes the sequential FAL installation steps in bar diagrams in a simple manner. To keep the algorithms simple, no differentiation between sequential and parallel working steps is made. All working steps are summed up to the AAP 'overall needed working time in FAL' as proposed in section 4.3.

This plugin has to be considered a prototype status to provide a simple example for extracting manufacturing information. In order to simplify the programming efforts, no model-to-model transformations from CDL to digital factory design languages [10] are made. Due to the modular plugin architecture of the CDL plugins, this plugin could be extended or replaced by model-to-model transformations, if a deeper involvement of manufacturing planning methods for cabin architecture analysis is needed.

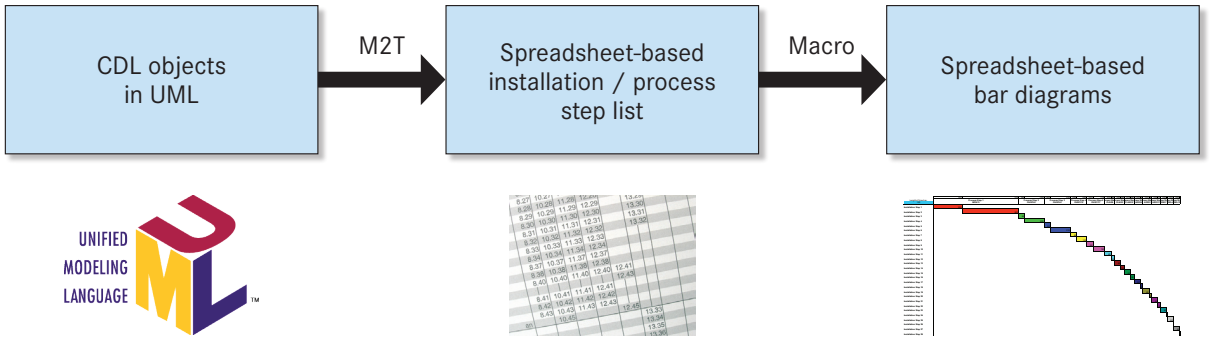

Figure 5.10: *Model-to-text transformation between CDL in UML and spreadsheet-based installation process bar diagrams*

5.1.7 Interface for Model Exchange

The plugin *cdl2export* (fig. 5.11) offers to select instances according to individual criteria with a GUI (fig. 5.12) and then creates 1:1 copies of these instances into a new UML model in a separate data file. There is no foreseen possibility to check the pragmatic sense of the model extract, as such checks require a clear purpose definition of the target model. At this stage, this cannot be described in a generic way.

Comparable to the previous plugin *cdl2fal*, this plugin has to be seen as prototype bridge to show the general feasibility of the requested multi-domain data exchange. Generally speaking, round trip methods are feasible, at least as long as there are no topological modifications within the model extract. This can enable consulting even further engineering groups with the data used by the CDL analysis model. For

instance, a potential link between CDL models and FEM can be set up using this plugin, but also cabin and aircraft systems models can be connected[100].

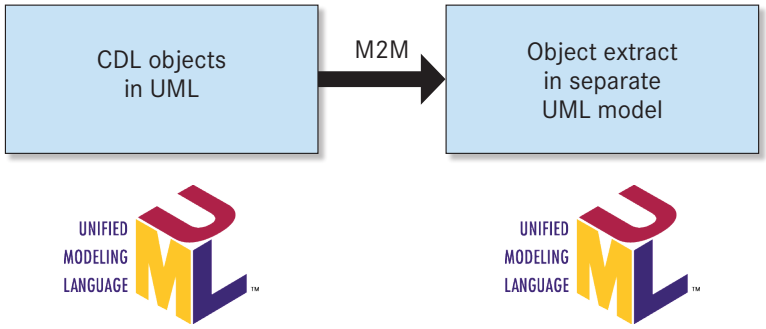

Figure 5.11: *Extraction of selected objects from a CDL model in UML*

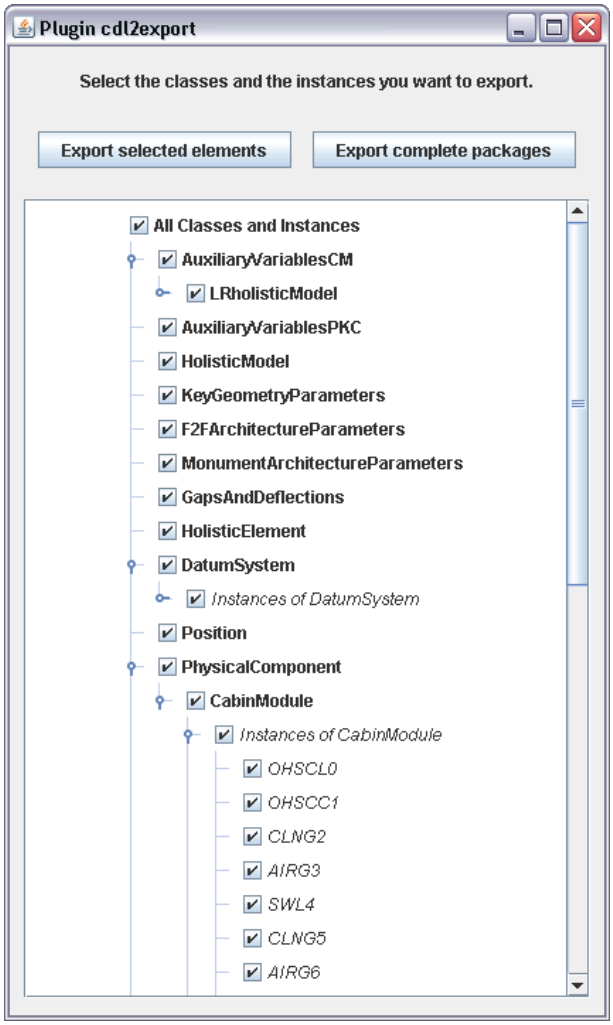

Figure 5.12: *GUI of the plugin* cdl2export *to select classes and instances for a data export*

[100] See subsection 6.2.2 for a more detailed discussion of these aspects.

5.2 Use Case Description

This section introduces a use case model named CDL_LR (cabin design languages for a long range aircraft). It contains some stowage bins, ceiling modules and sidewall modules of the constant cross section area including the corresponding attachment brackets, some crucial split lines as well as some structure components. Subject of investigations are three trade studies comprising several different technical scenarios, which will be compared against a baseline scenario (fig. 5.13). The trade studies aim at demonstrating the applicability range of the CDL_LR model for parameter and topology trade studies.

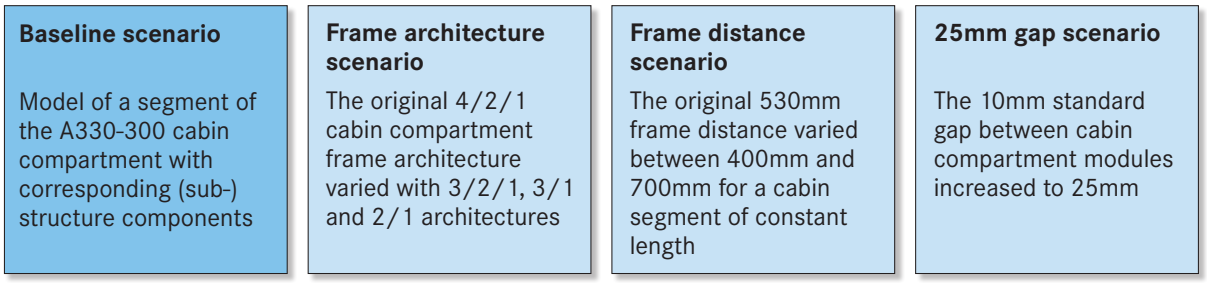

Figure 5.13: *The investigated cabin architecture scenarios within the CDL_LR use case*

Baseline scenario The baseline scenario is used as a reference for all successive comparative trades. It represents a segment of 15 frames length of a constant fuselage segment (see fig. 5.14). The baseline scenario uses a 4/2/1 frame architecture, which means that the stowage bins have a 4-frame length as a basis, followed by 2-frame or 1-frame modules to fill the chain gaps. For the lining panels, 2-frame (basis) and 1-frame modules are implemented. Figure 5.14 indicates the modeled cabin segment, while figure 5.15 shows the corresponding CAD data of the cabin modules and the structure components.

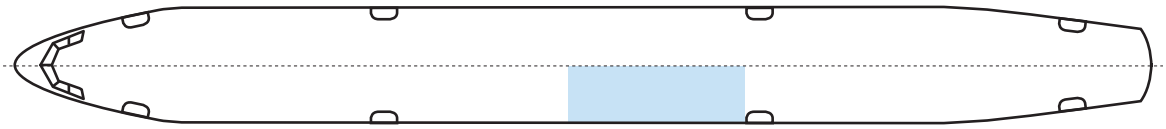

Figure 5.14: *The investigated cabin segment for the CDL_LR use case*

Trade study 'frame architecture' Besides the 4/2/1 frame architecture of the baseline scenario, the following alternative scenarios are investigated:

- A 3/2/1 frame architecture, where both the stowage bin modules and the lining modules have a 3-frame module as a basis. The full frame gaps at the cabin compartment chain ends are filled with either 2-frame or 1-frame modules.
- A 3/1 frame architecture, also with a 3-frame basis for all cabin compartment chain modules, but with only 1-frame modules to fill chain interruptions.
- A 2/1 frame architecture, where all cabin compartment chain modules have a 2-frame module as a basis and a 1-frame module as filler.

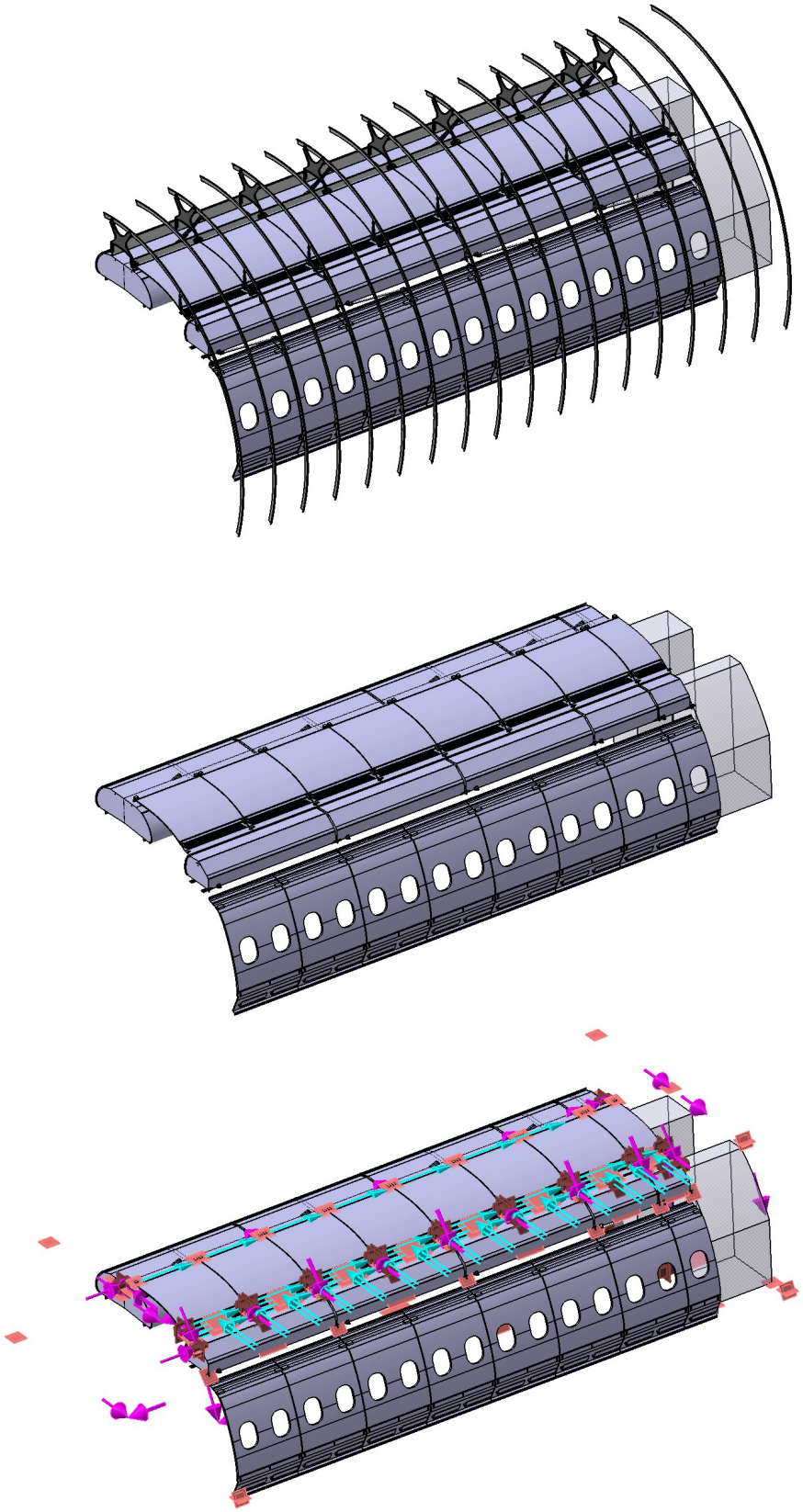

Figure 5.15: *DMU representation of the baseline scenario (top) – DMU with structure components hidden (middle) – DMU with loaded MECAmaster model (bottom)*

These trades aims at demonstrating the various repercussions of the different standard module frame lengths as examples for a topological trade study of the baseline scenario within multiple domains. Along with the full cabin module adaptations including interfaces, tolerances and weight, the geometry-related repercussions for the attachment brackets and for the substructure components concerning positioning and bracket mass need to be considered as well. However, secondary sizing and snowball effects for structure components such as for example the influence on primary structure mass, dynamic deflections and manufacturing-related topics are not modeled.

Trade study 'frame distance' The variation of the standard frame distance can be considered an example for an input parameter trade study with multi-domain repercussion. Starting from the baseline frame distance of 530mm, several scenarios between 400mm and 700mm are analyzed. Similar to the frame architecture trade study, no secondary effects for structure components are considered.

Trade study 'repercussions of a 25mm gap' The conceptual modification of the standard nominal gap width from 10mm to 25mm along with a tolerance optimization premise is an example for a multi-disciplinary iterative trade study. A combination of increased AKCs and MKCs enabling a similar level of the relative appearance PKC tolerances is searched, and the repercussions for installation PKCs are checked.

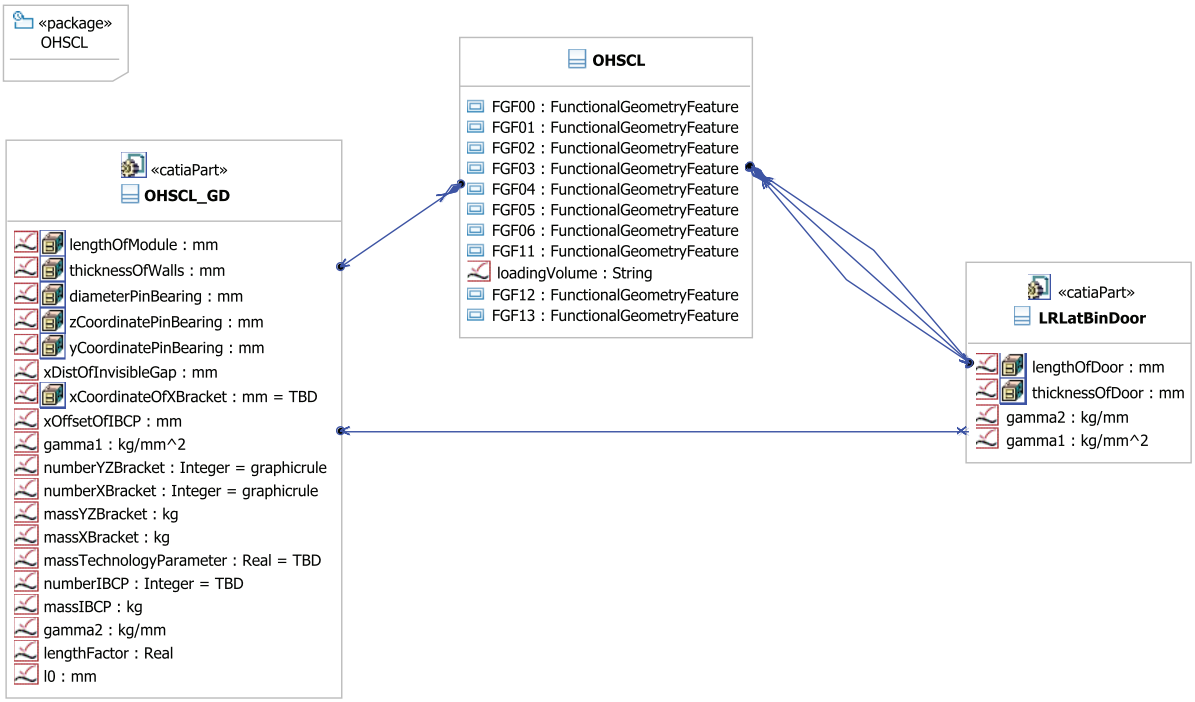

Figure 5.16: *Definition of the lateral stowage bin classes in the CDL_LR class diagram*

The case-specific CDL_LR class diagram[101] makes use of the UML generalization concept. The special CDL_LR classes inherit their functional behavior and their attributes from the generic CDL classes. Additionally, they are extended by individual features. To demonstrate the principle, figure 5.16 shows an example of a CDL_LR class for a stowage bin module called *OHSCL*. This class inherits from the

[101] Figure 5.17 provides a tabular overview about the content of the CDL_LR model. The full CDL_LR class diagram in UML can be seen in appendix C.

	Stowage bin modules		Ceiling modules		Sidewall modules			System components
	center	lateral	ceiling panel	air grid	light cover	window panel	dado panel	e.g., ATA21, ATA92
Cabin module-related classes								
Possible module frame architectures	4,3,2,1	4,3,2,1	3,2,1	3,2,1	3,2,1	3,2,1	3,2,1	-
3D geometry (simplified)	Y	Y	Y	Y	Y	Y	Y	N
Mass synthesis	Y	Y	Y	Y	Y	Y	Y	N
Cost synthesis	C	C	C	C	N	N	N	N
Definition of kinematic linkages	Y	Y	Y	Y	N	N	N	N
Definition of load interfaces	Y	Y	Y	Y	N	N	N	N
Definition of functional geometry features	Y	Y	Y	Y	N	N	N	N
Explicit definition of module datum	Y	Y	Y	Y	N	N	N	N
Synthesis of MKC tolerances	Y	Y	Y	Y	N	N	N	N
Definition of FAL installation sequence	Y	Y	Y	Y	N	N	N	N
Attachment brackets and related classes								
3D geometry (simplified)	Y	Y	Y	Y	N	N	N	N
Mass synthesis	Y	Y	Y	Y	N	N	N	N
Cost synthesis	C	C	C	C	N	N	N	N
Definition of kinematic linkages	Y_1	Y_1	Y_1	Y_1	N	N	N	N
Definition of load interfaces	Y_1	Y_1	Y_1	Y_1	N	N	N	N
Definition of functional geometry features	Y_1	Y_1	Y_1	Y_1	N	N	N	N
Synthesis of AKC tolerances	Y_1	Y_1	Y_1	Y_1	N	N	N	N
Linking of AKCs to integration datums	Y	Y	Y	Y	N	N	N	N
Airframe, substructures and related classes								
3D geometry (simplified)	Y_2							
Specification of integration datums	Y_3							

Y Available in CDL_LR model.
C Cost data and results not shown or discussed to keep information confidential.
N Not modeled.
Y_1 KinematicLinkages, LoadInterfaces, FunctionalGeometryFeatures and AKCs modeled between brackets and cabin modules. Corresponding objects between the brackets and the airframe and substructures not modeled according to the CDL philosophie.
Y_2 Airframe, substructure, and A-/B-brackets are only implemented as simplified 3D representation.
Y_3 Implemented integration datums: center substructure datum, lateral substructure datum, side shell datum, floor level datum.

Figure 5.17: *Overview of the content of the CDL_LR class diagram*

CDL class *CabinModule*. Some attributes and equations are appended, for instance for empirical parameterized mass estimations or for calculating the spatial location of related *FunctionalGeometryFeatures*. The individualized *SubComponent* classes have UML stereotypes [16] for CATIA V5 geometry visualization. The 3D data has been simplified and the calculation or estimation algorithms of the embedded design parameters are kept elementary.

As the overview table given with figure 5.17 shows, the model also contains a generic monument, which terminates the cabin compartment module chain. Concerning the airframe, aside from the attachment brackets mentioned and some substructure elements, the major fuselage structure components including the corresponding integration datums have their representations in the model. The implemented gap or split line types are shown in figure 5.18. Additionally, some installation PKCs are modeled.

PKC-A-001:	x-gap between two subsequent ceiling panels
PKC-A-002:	x-gap between two subsequent air grids
PKC-A-003:	y-gap between adjacent ceiling panels and air grids
PKC-A-004:	x-gap between two subsequent lateral stowage bins
PKC-A-005:	x-gap between two subsequent center stowage bins
PKC-A-006:	x-gap between the last lateral stowage bin and the subsequent generic monument
PKC-A-007:	y-flush between inboard edges of two subsequent ceiling panels
PKC-I-100:	Installation tolerance for up-long hole (2-way locator) of ceiling panel (not shown)
PKC-I-101:	Installation tolerance for up-long hole (2-way locator) of air grid (not shown)

Figure 5.18: *PKCs implemented in the use case*

Some cabin compartment aspects are omitted for the use case to reduce complexity. For instance, only the left hand side of the cabin segment is represented, since for the given level of granularity, the cabin compartment can be considered symmetrical. Furthermore, the modeled fuselage segment does not contain a door and entrance area and therefore neither the corresponding door and door frame lining modules nor the structure elements. Passenger seats are not implemented due to negligible influence in the mechanical architecture of the remaining cabin modules. For the generic monument only abstract geometry data and some tolerancing-related aspects are contained in the CDL_LR class diagram such that it is possible to simulate the interruption of the cabin compartment chain. Aircraft systems and system components are left out accordingly to focus on the mechanical integration of the cabin modules in advance. The influence of these simplifications on the architecture analysis task will be discussed in chapter 6. In order to keep information confidential, cost data and results will not be shown and discussed in this dissertation.

All classes and all design rules to compile the investigated scenarios build up the semantic hull of the CDL_LR model. The implemented rules will be discussed in detail subsequently in section 5.3.

5.3 Use Case Implementation

According to the concept formulated in section 4.4, the global rule sequence should be chosen such that it follows a tolerance management process. Therefore, the programs and Java rules of the CDL_LR root program[102] pay respect to the global rule sequence as shown in figure 5.19.

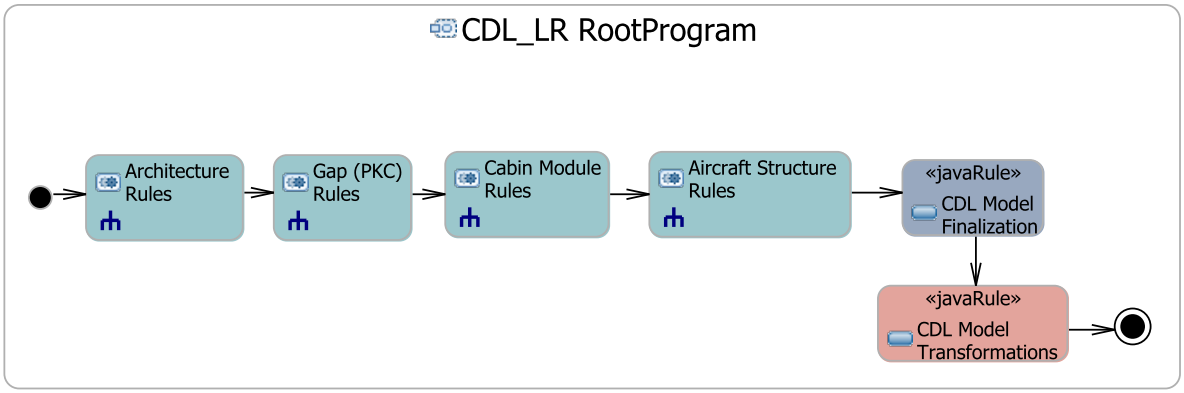

Figure 5.19: *The CDL_LR root program*

The first four programs of figure 5.19 consist of scenario-specific rules with graph-based model transformations for CDL vocabulary. The first two programs constitute a design and architecture framework (first phase according to section 4.4, described in sections 5.3.1 and 5.3.2), while the third and the fourth program build up the phase for the detailing of the model (second phase according to section 4.4, described in sections 5.3.3 and 5.3.4). The second but last step comprises the mentioned model checks (third phase), followed by the transformation of the CDL model into analysis models[103] (fourth phase), both summarized in section 5.3.5.

[102] The 'root program' is the top level of the rule sequence for graph-based design languages [75]. Consequently, a 'subprogram' or simply 'program' comprises a set of several rules and can be considered a 'program in a program', which can be re-executed every time it is referenced [10].

[103] The UML stereotype *«javaRule»* on the last two steps in fig. 5.19 indicates that they are realized by Java rules [75, 136].

5.3.1 Architecture Rules

According to the conventional design language concept, the first rule (green colored rule in fig. 5.20) is the ‘axiom’ [136], here called ‘Axiom LR’.

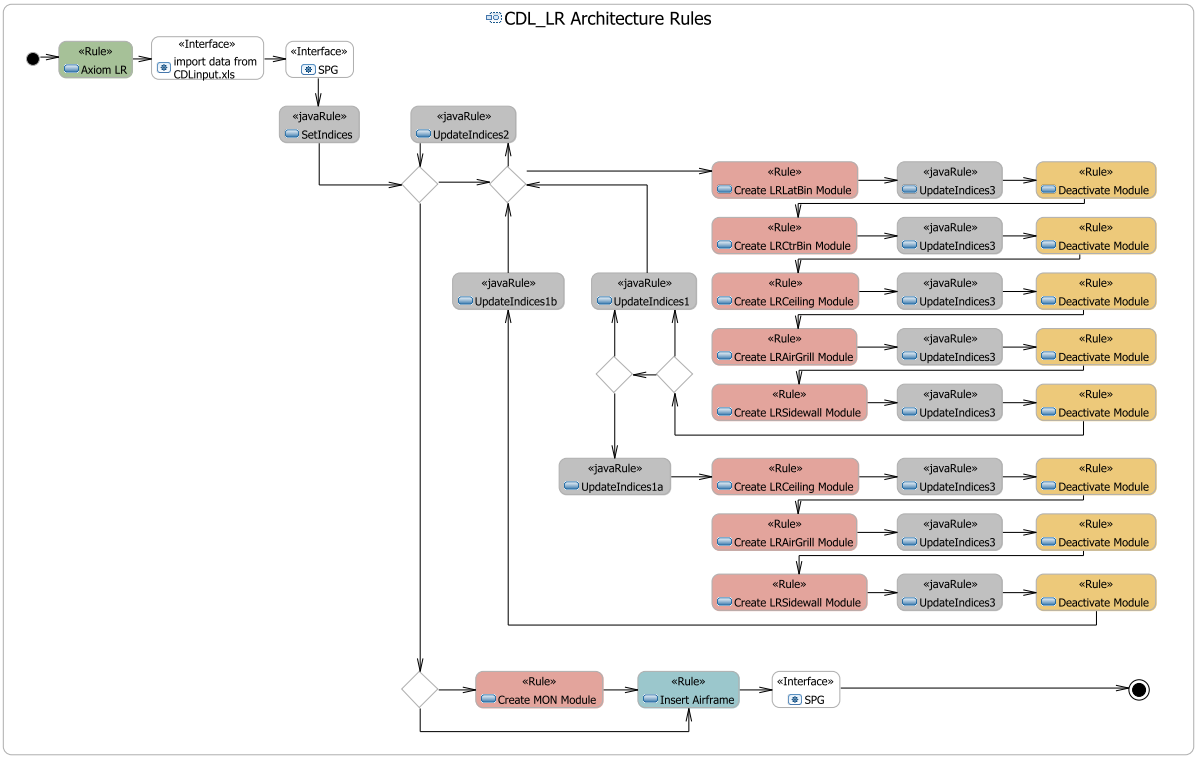

Figure 5.20: *The CDL_LR architecture rules*

This initial rule generates the instance ‘TopNode’ of the class *LRholisticModel*, which inherits from the generic CDL class *HolisticModel*. ‘TopNode’ owns several scenario-specific architecture input parameters as well as three index variables (*var1*, *var2* and *string1*), as figure 5.21 shows.

The architecture parameters are the main ‘adjustment screws’ for the various scenarios, which can be created with this design language model. The first input parameters are the name and the aircraft program of the scenario along with a trade number to distinguish between the different trades. Among the major technical input parameters are the frame distance of the fuselage section, some specific geometric dimensions like the y- and z-coordinates of the floor level and of the crown area, as well as the length of the cabin compartment segment. The cabin compartment module frame length architecture is defined by two parameters. The different gap or split line types between loaded and unloaded cabin modules both get a definition of the nominal gap size as well as of the required minimum gap size to cover in-flight deflection movements.

The user can input the parameters in a spreadsheet-based input mask (see fig. 5.22) instead of changing them directly in the rule editor window of the rule ‘Axiom LR’. This offers the possibility to create multiple input files for different scenarios in advance, which can be compiled and executed one after another – for example by the support of batch files. The import of the parameter values is performed directly after the rule ‘Axiom LR’, followed by a first calculation of the derived model parameters (first two white rules in fig. 5.20).

Figure 5.21: *Graphical representation of the initial rule ('Axiom LR')*

UML Model Naming		
A/C Program	ACProgram	Long Range
Name of Model	ModelName	Baseline
Trade Number	TradeNumber	9000
Key Geometry Parameters		
Frame Distance	FrameDist	530
Z-Height of Floor Level	ZheightFloorLevel	0
Y-Distance of Meeting Point between Floor Level and Side Shell	YrefFloorLevel	2705
Z-Height of CTR Crown Area Z-Reference	ZrefCrownArea	1927,3
Y-Distance of Meeting Point between CTR Crown Area and Side Shell	YrefCrownArea	1785,8
F2F Architecture Parameters		
F2F length of module chain in frame bays	F2FchainFrameLength	6
F2F Frame Architecture 1	F2FframeArchitecture1	4
F2F Frame Architecture 2	F2FframeArchitecture2	2
Monument Architecture Parameters		
Monument length in frame bays	MonumentFrameLength	2
Tolerancing, Gaps and Deflections		
Unloaded to unloaded Component nominal gap width	NominalGapUnloaded2Unloaded	10
Unloaded to unloaded Component minimal gap width (in-flight deflections)	MinimalGapUnloaded2Unloaded	5
Loaded to unloaded Component nominal gap width	NominalGapLoaded2Unloaded	25
Loaded to unloaded Component minimal gap width (in-flight deflections)	MinimalGapLoaded2Unloaded	17
Loaded to Loaded Component nominal gap width	NominalGapLoaded2Loaded	25
Loaded to Loaded Component minimal gap width (in-flight deflections)	MinimalGapLoaded2Loaded	17
Name of Global Section Datum	GlobalSectionDatum	FuselageSection
Standard Interface Tolerance	stdIFtol	0,25
Standard Tooling Tolerance	stdToolTol	0,8

Figure 5.22: *Spreadsheet-based input for the CDL_LR architecture parameters*

Now follows a large loop of rules in order to insert the cabin module instances. The actual architectural rules to insert the cabin compartment chain modules into the fuselage segment can be expressed *verbally* as follows:

> *Fill the cabin segment with the length of F2FchainFrameLength frame bays with the largest possible module size (F2FframeArchitecture1) starting at the first frame.*
>
> *Once the largest module size no longer fit into the remaining gap, use the next possible module size (F2FframeArchitecture2).*
>
> *Once this size no longer fits into the remaining gap, reduce the module size by 1 and continue to fill in as much as possible. Repeat this procedure until the segment is completely filled.*
>
> *Consider that the largest module size for the lining panels is 3 frame length, while stowage bins can have a length up to 4 frames. This means that if F2FframeArchitecture1 is set to 4, the lining panels need to start with the length as specified in F2FframeArchitecture2 ('lining panel maximal length exception').*

The correspondingly chosen design language rule algorithm reads as follows:

1. The Java rule 'SetIndices' (first rule marked in gray, see fig. 5.20) sets the index variable *var1* to the frame length of the cabin compartment segment (F2FchainFrameLength) and *var2* to the first frame length architecture parameter (*F2FframeArchitecture1*).

2. The following two decision nodes check if $var2 > 0$ and then if $var1 - var2 \geq 0$.

   (a) If $var1 - var2 < 0$, *var2* is reduced to the next frame architecture length (*F2FframeArchitecture2* for the first time, $var2 - 1$ for the other cases).

   (b) If $var2 = 0$, the loop quits.

3. If the conditions from step 2 are fulfilled, a loop to insert the cabin modules is entered. Within the loop, for each *CabinModule* to be insert, three rule steps are followed:

   (a) Within the rules named 'Create *X* Module' (red rules in fig. 5.20), the LHS searches for the instance 'TopNode' classified by *LRholisticModel*[104] (fig. 5.23). On the RHS, a specific cabin module instance with all slots is inserted into the model and is linked to 'TopNode'. The slot *inWork* is flagged as 'true', which means that the instance can be considered as activated. In addition, the slot *isLoaded* is set according to the module's load behavior.

   (b) The following Java rule 'UpdateIndices3' sets the slot *FWDframePosition* of the activated cabin module instance to *F2FchainFrameLength - var1*. The slot *lengthInFrames* is set to *var2*, if *var2* equals or is smaller than the maximal allowed module length. Otherwise, *var2* is set to *F2FframeArchitecture2*, which is the next valid module frame length in this case.

   (c) The rule 'Deactivate Module' (orange rules in fig. 5.20) searches for all *CabinModule* instances whose slot *inWork* is 'true' and sets the slots to 'false' (see fig. 5.24).

4. Once all *CabinModule* instances of the loop are inserted and positioned, two decision nodes check whether the 'lining panel maximal length exception' occurred. If so, a Java rule called 'UpdateIndices1a' reduces the value of *var1* by *F2FframeArchitecture2* and repeats the rules 3a through 3c for the lining panels.

5. After that, the loop goes again to the decision nodes as described in step 2.

[104] Since only one instance of the classifier *LRholisticModel* exists, the rule shown in fig. 5.23 searches for 'TopNode'.

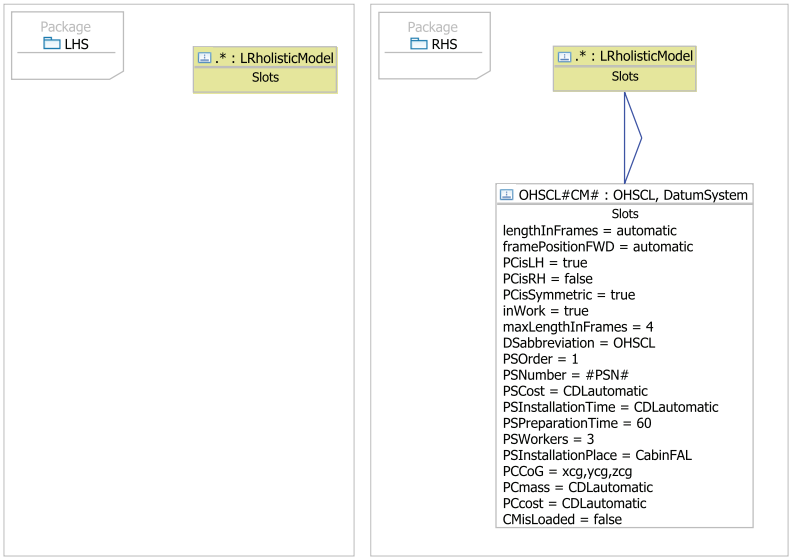

Figure 5.23: *Rule 'Create LRLatBin Module' to insert a lateral stowage bin module*

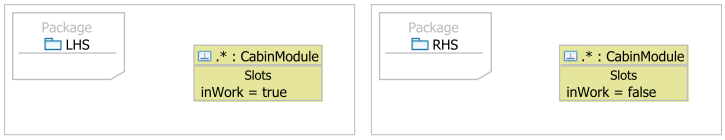

Figure 5.24: *Rule 'Deactivate Cabin Module' to deactivate a cabin module*

The rule 'Create MON Module' (last red rule in fig. 5.20) inserts the generic monument, if the user opted for a monument within the architecture parameter input mask. The rule is similar to the previous 'Create *X* Module' rules. Again, a decision node performs the corresponding check.

The rule 'Insert Airframe' (light blue rule in fig. 5.20) is the last architecture rule. It inserts the *StructureComponents* for the global section datum called 'Fuselage Section' and for the cabin integration datums called 'FloorLevel', 'SideShellLH', 'LATSubstructureLH' and 'CTRSubstructure' (fig. 5.25). To simplify the data model, the right-hand datums are not modeled. The tolerances in global x-/y-/z-direction as well as the coordinates of the virtual linkages between these datums are defined within slots inside the *StructureComponent* instances. The virtual linkage coordinates depend on the architecture geometry parameters and are to be calculated subsequently.

After the execution of the architecture rules, the design graph (see fig. 5.26 on next page) does not yet have any loops or complex tree structure[105]. All *PhysicalComponents* are linked back to the instance 'TopNode' (red bullet in the center of fig. 5.26 on next page) to have access to the architecture geometry parameters.

[105]Compare fig. 5.26 with the design graph in fig. 1.10 (page 15) containing one loop due to a modeled tolerance stack.

Figure 5.25: *Rule 'Insert Airframe' to insert structure component instances*

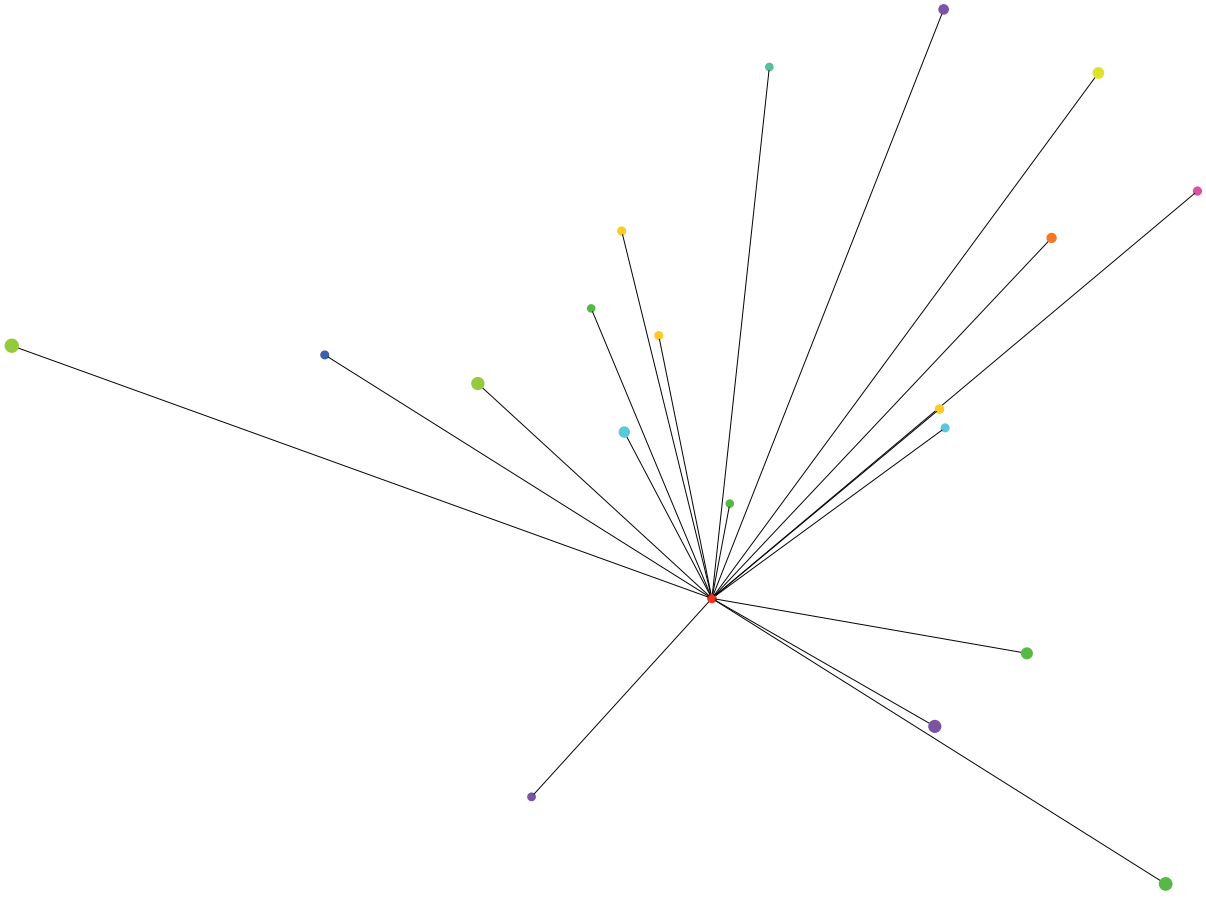

Figure 5.26: *Design graph of the baseline scenario after the execution of the architecture rules*

5.3.2 Gap Rules

Verbally expressed, the generic rule for the *PKC* implementation into the model reads as following:

> *If a gap or a split line is important for the overall cabin quality and therefore leads to possible requirement repercussions for installability and manufacturability aspects, it needs to be modeled explicitly.*

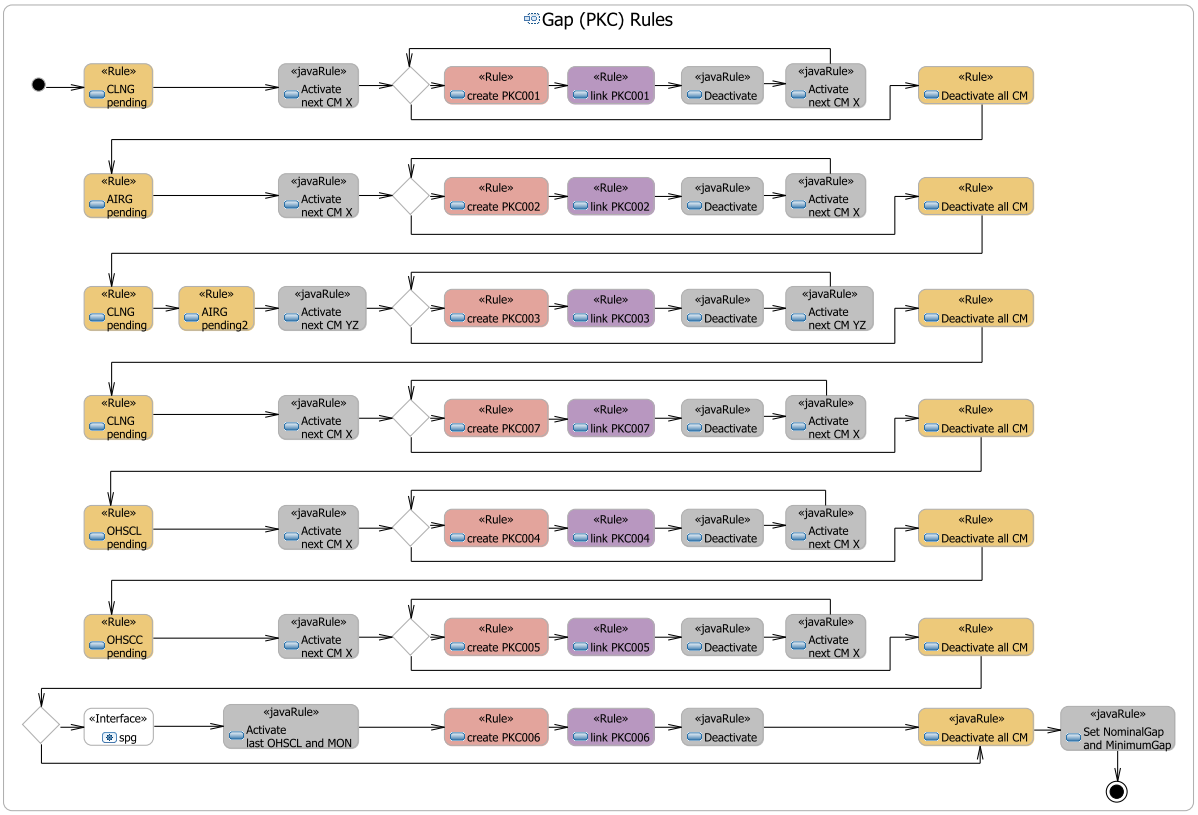

Figure 5.27: *The CDL_LR gap (PKC) rules*

The gaps and split lines as mentioned in figure 5.18 in section 5.2 match this classification. For this reason, the following algorithm is run through (fig. 5.27):

1. Before entering the first loop, the affected Cabin Modules are set to ‘pending’ using a graphical rule (rule ‘X pending’, see orange rules in the first column in figure 5.27).

2. After that, the Java rule ‘Activate next CM X’ searches for two directly subsequent *CabinModules* within those set to ‘pending’. Due to the generic set up, the rule can be used for any activation task independent from the explicit type of the *CabinModule* classifier. The routine grabs the first *CabinModule* of the chain and checks if its slot *framePositionAFT* has the same value as the slot *framePositionFWD* of a second candidate. The search compares any pair of pending *CabinModules* until either a valid pair is found or until the end of the list has been reached without a further finding. If a valid pair is found, the two candidates are activated by setting the slot *inWork* of the forward module to ‘true1’ or the one of the aft module to ‘true2’.

3. A decision node now checks if there are any activated *CabinModules*. If so, a loop is entered:

   (a) Within the loop, at first a new *PKC* instance is inserted into the model and activated (red rules in fig. 5.27), as figure 5.28 on the next page shows for the example of the x-gap between two adjacent ceiling panels (PKC-001).

   (b) Now this newly inserted instance has to be linked to the corresponding *CabinModules* (violet rules in fig. 5.27). The rule 'Insert PKC-X' therefore searches for the two activated *CabinModules* and for the activated *PKC* instance on the LHS. On the RHS, two new instances are inserted and are classified as *FunctionalGeometryFeature* and as *MKC*. The coordinates and the tolerances either are specified manually in the rule or are calculated using formulas which are part of the corresponding case-specific class definition.
   In the specific example of PKC-001, the *FunctionalGeometryFeature* and *MKC* instances represent dedicated points at the edge of the ceiling panels with a tolerance in x-direction (fig. 5.29 on the next page) which is synthesized using module parameters. The datum system itself has not yet been modeled explicitly. However, as the *CabinModule* instance is always classified Datum System as well, there is no definition gap at this stage: up to now, the datum definition is implicit.

   (c) The Java rule 'Deactivate' (gray rule in fig. 5.27) deactivates the *PKC* and the forward *CabinModule* instance. The aft *CabinModule* is set back to 'pending', as it could still be part of a new pair of *CabinModules* between which a gap needs to be inserted.

   (d) The rule 'Activate next CM X' (gray rule) is called again to try to activate the next pair of *CabinModules*. It leads back to the initial decision node.

4. Once the loop is left, all *CabinModules* are set inactive.

5. Steps 3 and 4 are executed in a similar way for each *PKC* type.

6. As second to last step within the gap rules program, there is a decision node to check whether the optional monument module is in the model. If so, one further *PKC* is implemented representing the x-gap between the last lateral overhead stowage bin and the monument in a similar way as described with steps 1 through 4 above.

7. The gap rules program is finished by a Java rule named 'Set NominalGap and MinimumGap'. For each *PKC* instance it checks, if the corresponding *CabinModules* are loaded modules by reading the value of the slot *isLoaded*. The nominal gap size and the minimal gap size of each *PKC* are thus set corresponding to the architecture parameter definitions.

After these steps, the design graph now contains loops, as can be seen in figure 5.30[106] (see next page). These are the graphical representation of the over-constraining tolerance requirements SÖDERBERG already describes as axiomatically coupled tolerance chains [148].

[106] Compare this design graph with the one representing the model after the architecture rules shown in fig. 5.26 (page 95). The red bullet in the center is the instance 'TopNode' containing the architecture parameters and thus linking all model objects.

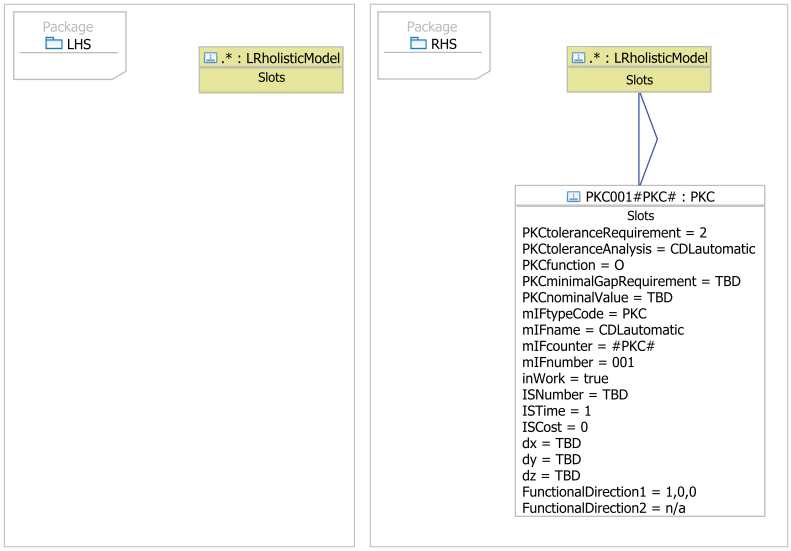

Figure 5.28: *Rule 'Create PKC-001' to create a PKC instance*

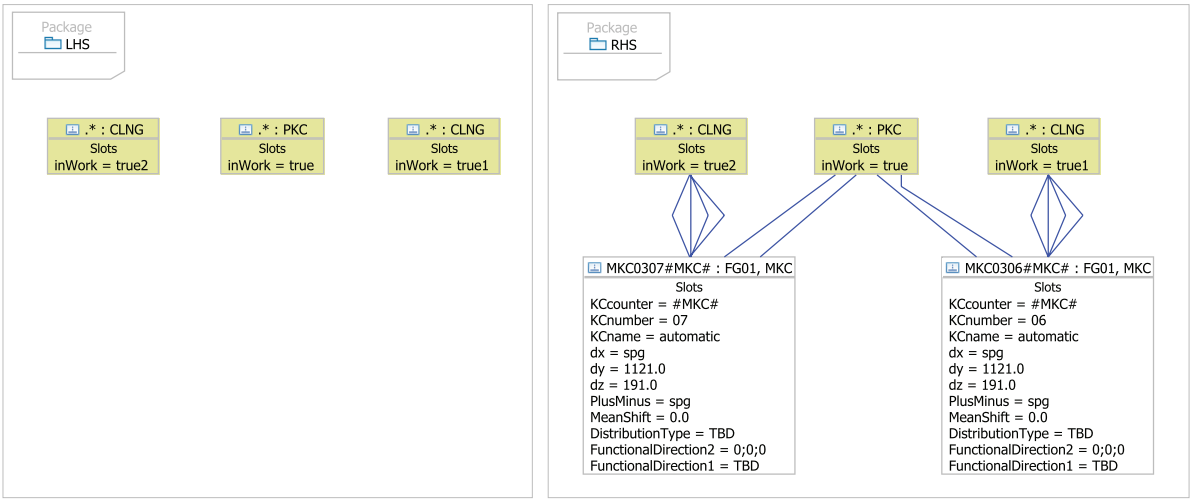

Figure 5.29: *Rule 'Insert PKC001' to link a PKC instance with two ceiling panels*

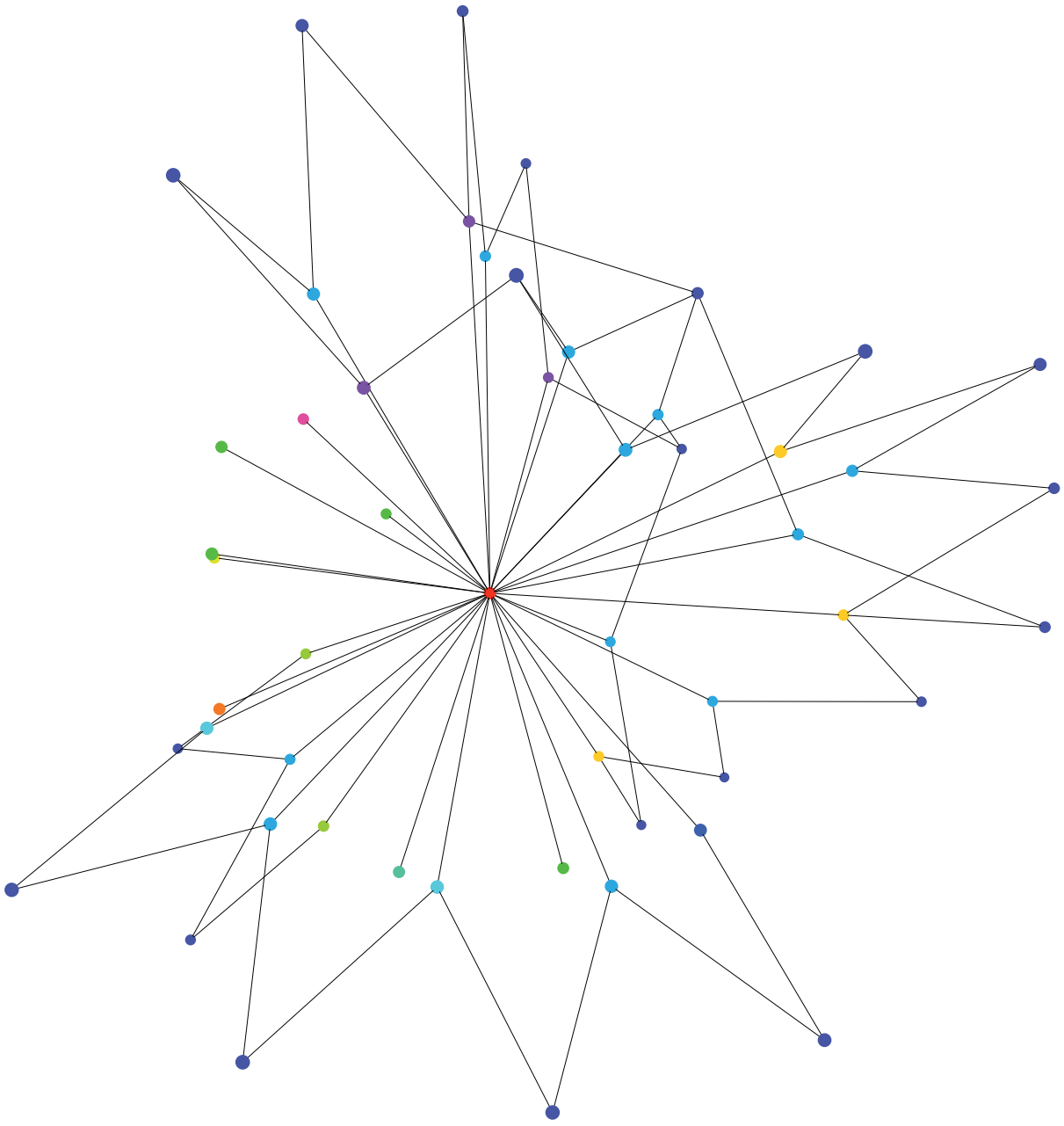

Figure 5.30: *Design graph of the baseline scenario after the execution of the gap (PKC) rules*

5.3.3 Cabin Module Rules

The purpose of the cabin module rules (fig. 5.31) is to enrich the model with the following aspects:

- 3D geometry for the visualization of the *CabinModule* instances
- Mass estimation methods
- *FunctionalGeometryFeatures* of the *CabinModules* for the explicit definition of the cabin modules' datum system or for tolerance definitions, for the definition of the kinematic linkage system (as far as possible) and for the definition of the load interface system (as far as possible)
- *FunctionalGeometryFeatures* on the aircraft-side as a counterpart for *MechanicalInterfaces* including tolerances linked to the corresponding integration datum system (as far as possible)
- Brackets on aircraft-side (as far as possible)

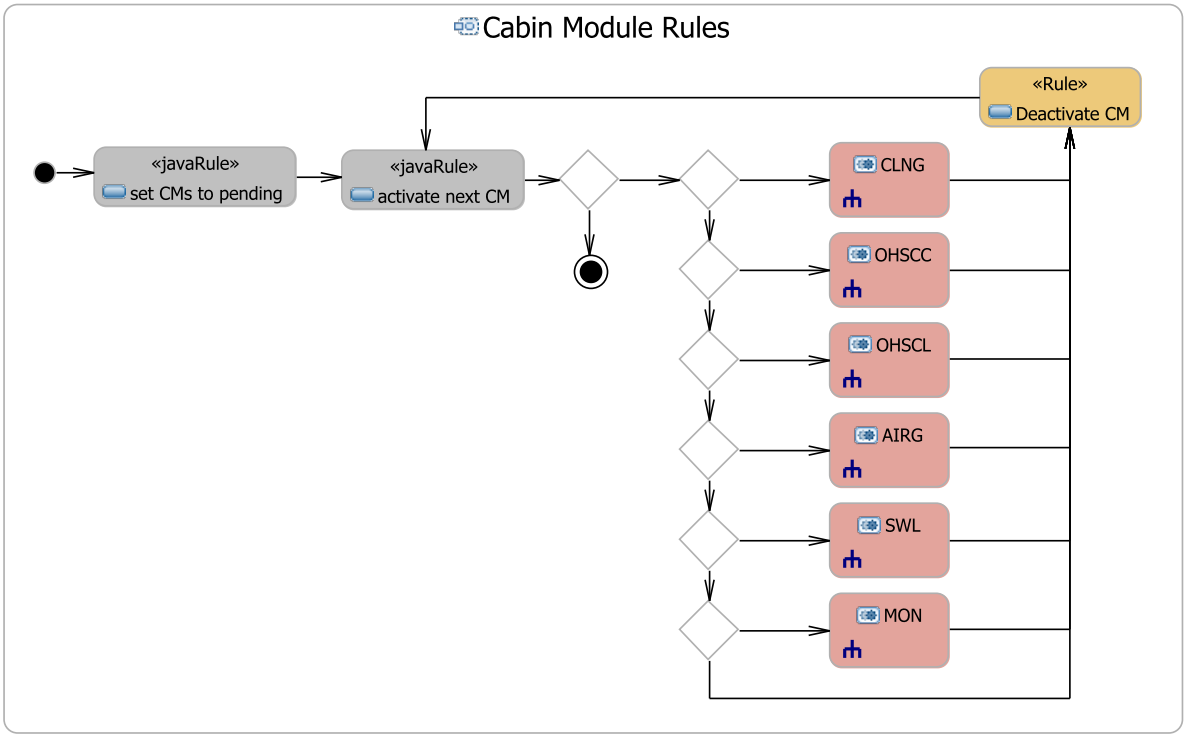

Figure 5.31: *The CDL_LR cabin module rules*

The cabin modules program starts with a rule setting all *CabinModules* to 'pending' (first gray rule in fig. 5.31). A loop follows, at the beginning of which a pending Cabin Module instance is activated. Within the loop, every *CabinModule* instance is processed with a subprogram corresponding to the cabin module's class specification. In this loop the datum system will be made explicit and the linkage system will be deployed. Generally, it can be said that completely generic definitions of the datum and linkage systems are difficult to realize and are not pragmatical. But it is possible to create linkages in a KBE manner individually per cabin module.

To demonstrate the principle, the cabin module rules for the lateral overhead stowage bin detailing process are shown in figure 5.32 and are subsequently described in detail.

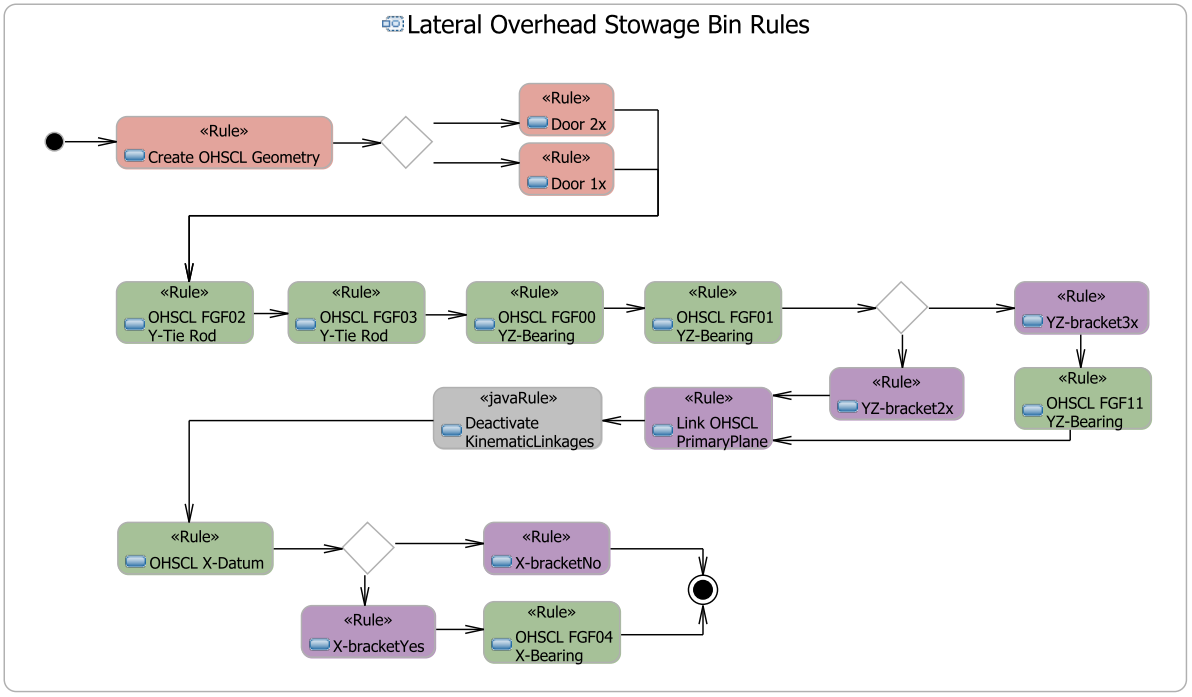

Figure 5.32: *Design rules for the lateral overhead stowage bin module*

The first red rule in figure 5.32 called 'Create OHSCL Geometry' is a graphical rule for enriching the *CabinModule* with a linked *SubComponent* instance classified by *OHSCL_GD* with a UML stereotype for DMU visualization purposes (fig. 5.33). Some module-internal geometry parameters (*numberYZBracket*, *numberXBracket*) are not yet set, as they depend on the characteristic frame length of the respective module. The number of the stowage bin doors depends on the individual module's frame length. A decision node checks for the value and leads to the corresponding rule 'Door 1x' to insert either one door (for bin lengths up to 2 frames) or two doors (rule 'Door 2x').

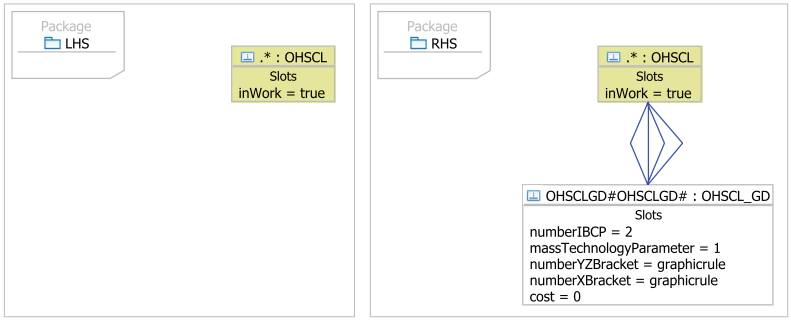

Figure 5.33: *Rules to insert visualization geometry data for a lateral overhead stowage bin*

The following four green rules (see fig. 5.31) serve the purpose to link the *CabinModule* instance with *FunctionalGeometryFeatures*. In contrast to the gap rules, the *FunctionalGeometryFeatures* now reflect attachment interface locations – instead of gap or PKC locations – and are linked to *MechanicalInterface* instances classified as both *KinematicLinkages* and *LoadInterfaces*. Figure 5.34 shows an example for a

rule inserting a y-tie rod attachment between the stowage bin and the aircraft structure. It can be seen, that the *FunctionalGeometryFeature* of the stowage bin module is directly linked to a specific *MechanicalInterface* instance representing one lower y-attachment interface (fig. 5.34). The *MechanicalInterface* is linked to a second *FunctionalGeometryFeature* belonging to the *StructureComponent* instance 'LAT-Substructure' acting as integration datum. The two y-/z-interfaces are implemented accordingly.

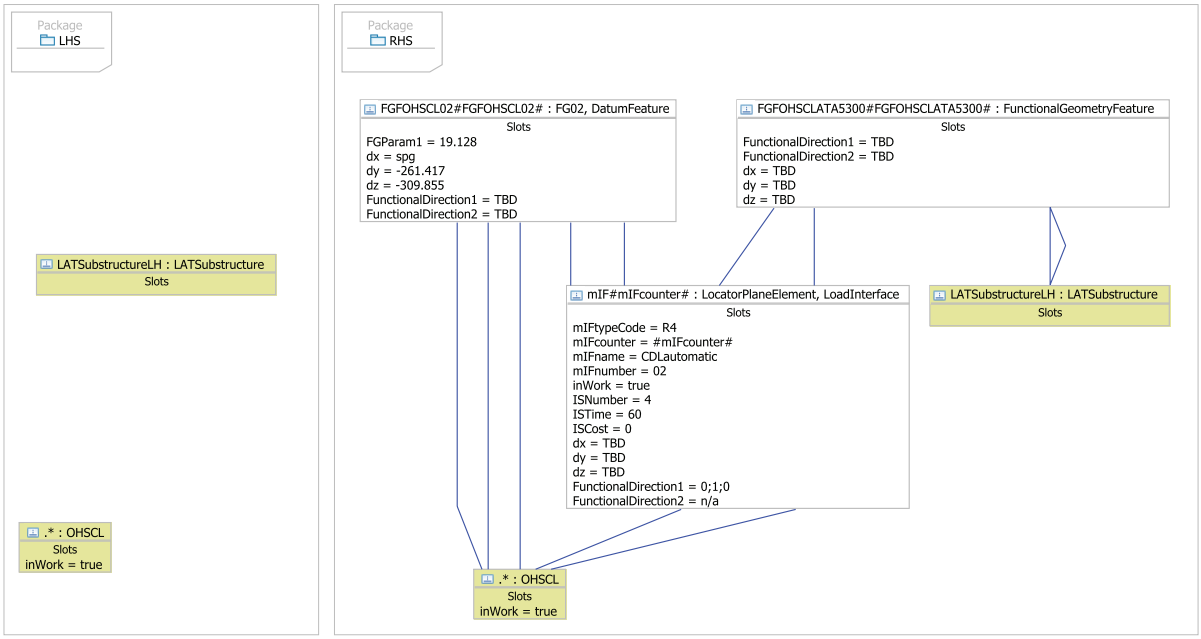

Figure 5.34: *Rules to insert functional geometry features data for a lateral overhead stowage bin*

The *FunctionalGeometryFeatures* created in the first four green rules (fig. 5.32) are all classified as *DatumFeatures*. All together, they constitute the explicit datum system of the *CabinModule*. The z-datum features are the two *YZ-Bearings*, which constitute the secondary datum. Together with the two lower y-tie rod attachment holes, they are also y-datum features and altogether act as primary datum plane. The stowage bin's x-datum is represented by a special *FunctionalGeometryFeature*, which is not automatically linked to any *MechanicalInterface*, and is inserted by the rule 'OHSCL X-Datum' (fig. 5.35).

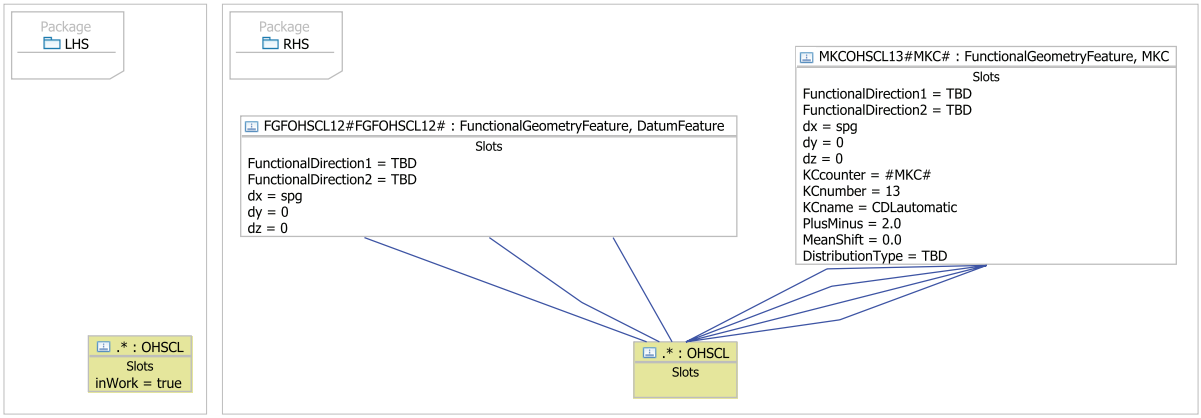

Figure 5.35: *Rules to insert the x-datum feature for a lateral overhead stowage bin*

In parallel to the explicit definition of the module's datum system, the kinematic linkage system and the load interface system also develop step by step with the green rules. The special *lateral stowage bin linkage systems* is used here. Therefore the *MechanicalInterfaces* are classified as *KinematicLinkages*[107] and also as *LoadInterfaces* (fig. 5.34). These interfaces fulfill both the locating functions as well as the fixation functions – in y-/z-direction in the case of the YZ-Bracket, in y-direction in the case of the y-tie rod attachment. When the instances are created, their slot *inWork* is set to 'true'. These four interfaces lock five DOF of the stowage bin. The missing locating function in x-direction depends on the relative position of the stowage bin in the chain. The x-locating *MechanicalInterface* thus has to be inserted later with the structure subcomponent rules.

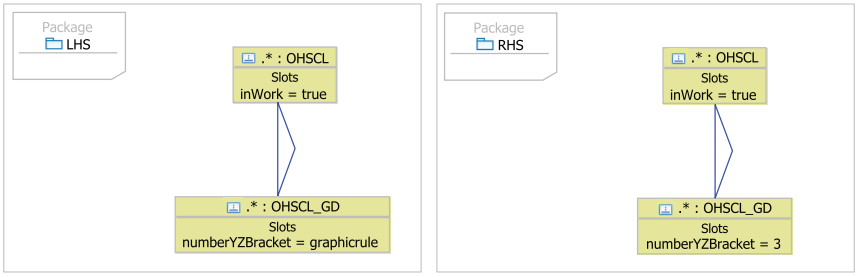

Figure 5.36: *Rules to set the value of the slot* numberYZBracket *of a lateral overhead stowage bin*

The fifth green rule in figure 5.32 has a decision node put in front which checks for the frame length of the specific module. The subsequent violet rule assigns the correct value to the slot *numberYZBracket* (fig. 5.36). If the stowage bin has the length of four frames, a third y-/z-interface and a x-tie rod interface are implemented, which are classified as load interface only.

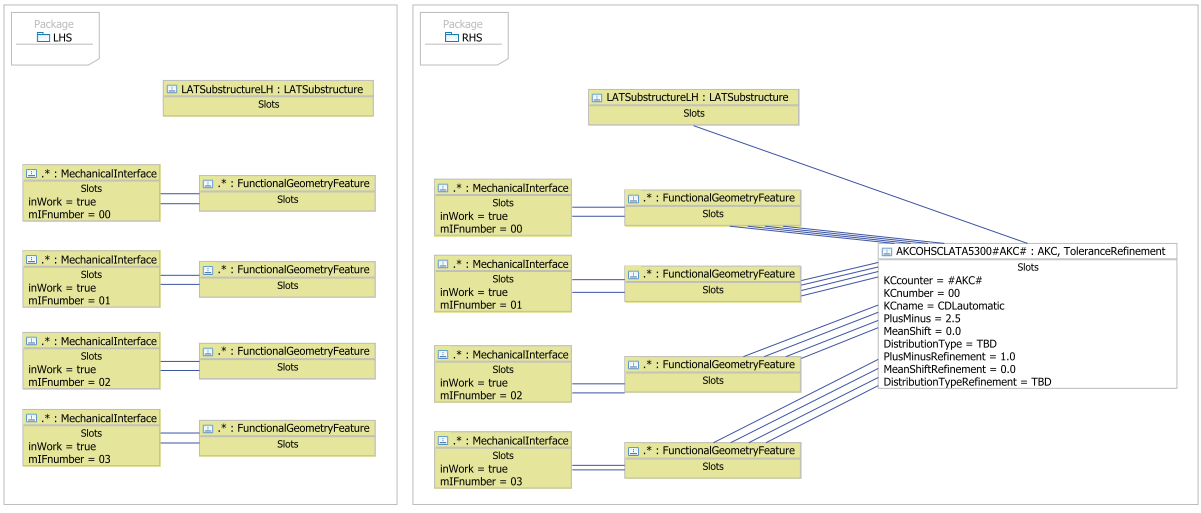

Figure 5.37: *Rules for linking locating interfaces of a lateral overhead stowage bin to one common AKC tolerance*

[107] To be precise, in the example of fig. 5.34, the *MechanicalInterface* is classified as a special *KinematicLinkage*, a *LocatorPlaneElement*. The *MechanicalInterface* of the *YZ-Brackets* is classified as *Rotated4WayLocator*.

With the rule ‘Link OHSCL PrimaryPlane’, the Functional Geometry Features on the aircraft side of all four mentioned locating interfaces are linked together to one AKC group tolerance including a group tolerance refinement. This coupling will later show its effect in the context of tolerance analysis calculations. On the LHS of the named rule, the four involved active *MechanicalInterfaces*, the corresponding *FunctionalGeometryFeatures* on the aircraft side and the ‘LATSubstructureLH’ instance are searched. On the RHS, the AKC tolerance[108] is inserted and linked to the four *FunctionalGeometryFeatures* and to the ‘LATSubstructureLH’ as the corresponding datum (fig. 5.37).

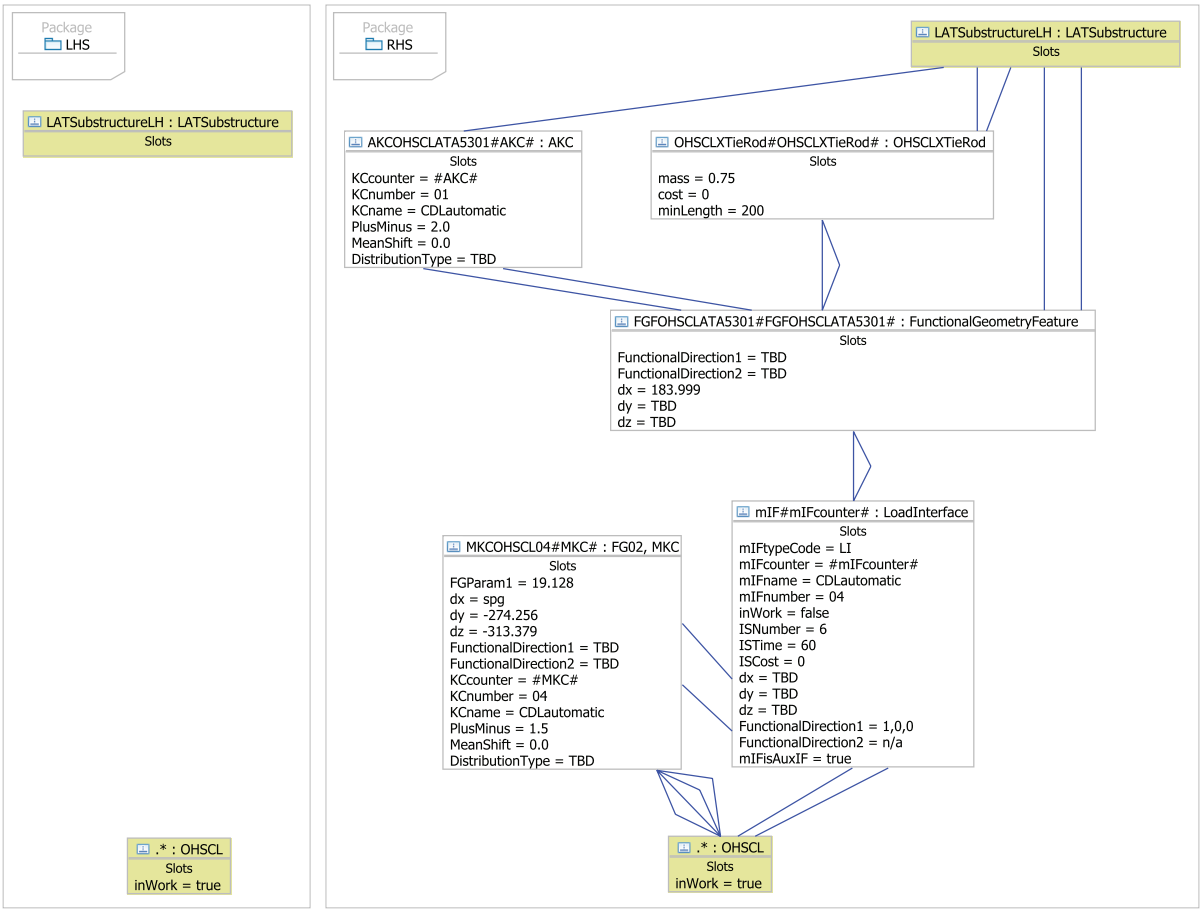

Figure 5.38: *Rules for inserting functional geometry features for a lateral overhead stowage bin*

Where possible, *FunctionalGeometryFeatures* belonging to the aircraft structure are directly linked with brackets (*StructureSubcomponent* instances) resembling the physical counterpart on the aircraft side. The rule discussed for the y-tie rod shown in figure 5.34 as well as those for the *YZ-Bearing* do not yet implement such a bracket instance. The reason is that the same bracket is needed for another stowage bin module, and therefore should only be created and linked after the other cabin module instances exist. However, at some places it is possible: figure 5.38 shows the rule ‘OHSCL FGF04 X-Bearing’ for inserting and linking a *MechanicalInterface* for the x-tie rod (also see fig. 5.32). As this bracket fixes one single stowage bin module, the corresponding bracket instance can be inserted and linked to the corresponding *FunctionalGeometryFeature* directly here.

[108] For the AKCs constant tolerance values are used. Alternatively, synthesis equations could be implemented to calculate the AKC tolerances depending on design parameters, as it is done for the MKC tolerances.

Within the design graph, further loops are established between the *CabinModules*, the *KinematicLinkages* or the *LoadLinkages* and the corresponding integration datum[109] (fig. 5.39). While the cycles including the *PKC* instances reflected the coupled tolerance chains, these cycles here resemble the linkage system. In the particular case of the lateral stowage bin, the loops are even coupled. This means, they are parametrically depending on each other, because the *FunctionalGeometryFeatures* on the aircraft side of the y- and z-locating *MechanicalInterfaces* constitute a tolerance group as described above.

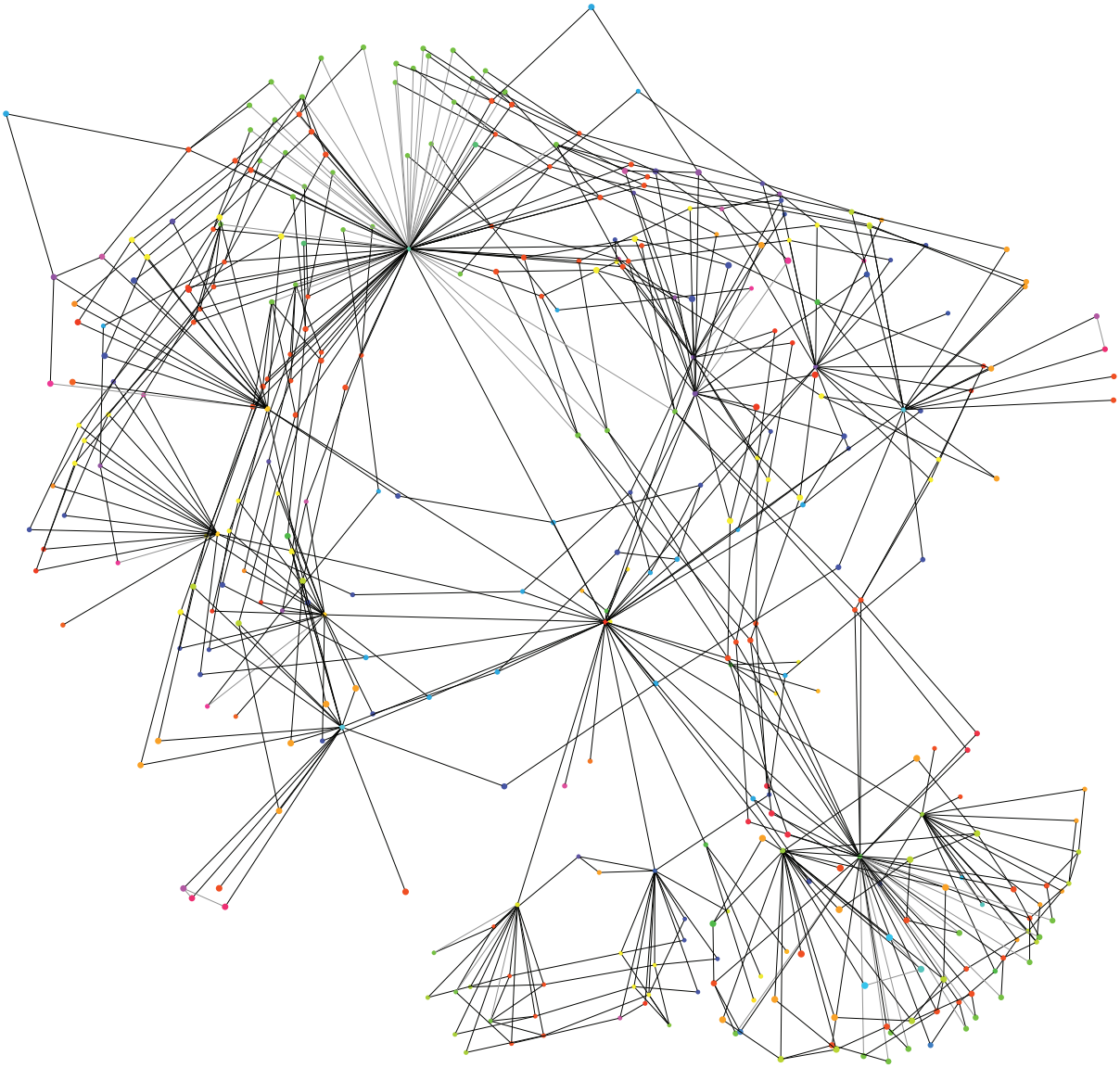

Figure 5.39: *Design graph of the baseline scenario after the execution of the cabin module rules*

[109] Compare with the design graph after the execution of the architecture rules (fig. 5.26, page 95) or after the gap (PKC) rules (fig. 5.30, page 99) respectively. The increasingly comprehensive design graph indicates the complexity of the data model.

5.3.4 Aircraft Structure Rules

The aircraft structure rules (fig. 5.40) complement the cabin module rules. These rules have the task of inserting any required but open aspect into the model such as

- implementing 3D visualization data of the section frames
- finalizing the definition of the mechanical interfaces[110] for the *CabinModules*
- finalizing the implementation of *FunctionalGeometryFeatures* on the aircraft-side including *AKC* tolerances and including associations to *KinematicLinkages*
- implementing the brackets (*SubComponents* of the *StructureComponents*) on the aircraft-side.

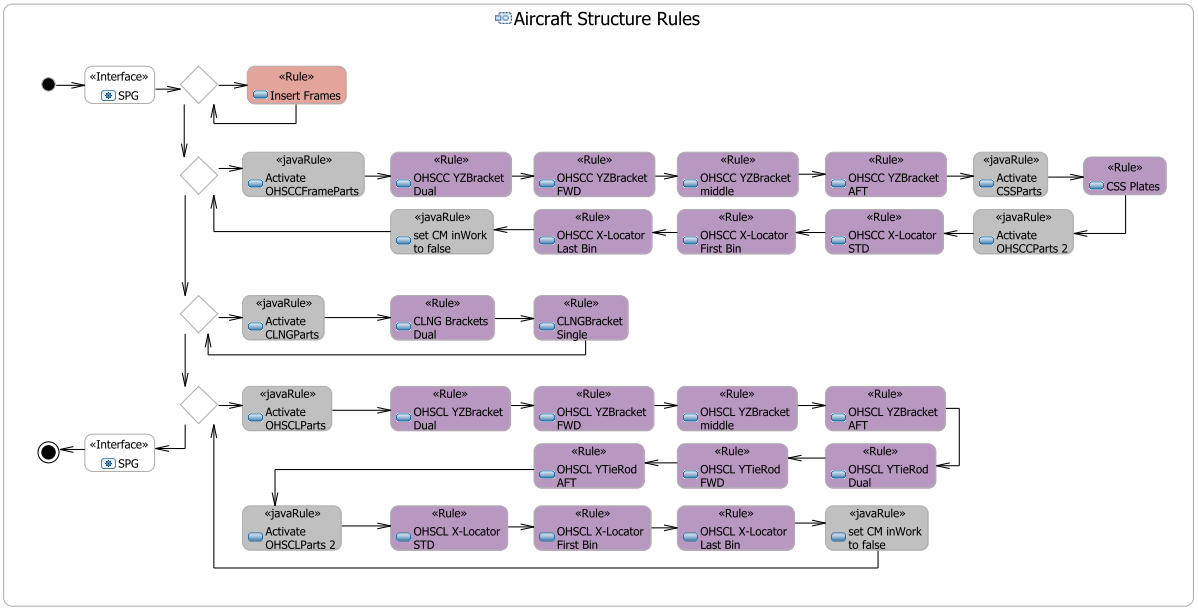

Figure 5.40: *The CDL_LR aircraft structure rules*

In some cases the brackets and their *FunctionalGeometryFeatures* cannot be assigned directly to one single *CabinModule*, for instance if a bracket or parts of it are multi-functional attachment brackets to locate and fix several parts. For these cases, the structure components program fulfills the same task as the cabin modules program and inserts and links the missing *MechanicalInterfaces*, *FunctionalGeometryFeatures* and *SubComponents* or brackets on the structure-side.

The difference from the cabin module rules is that for the aircraft structure rules the iteration runs over the fuselage frames using the frame number instead of iterating over the *CabinModule* instances. In particular, this is required for the center substructure for the attachment of the center stowage bins, for the lateral stowage bin attachment brackets as well as for the ceiling panel and air grid attachment brackets.

[110] In particular this means the finalization of the definition of the kinematic linkage system and of the load interface system for each *CabinModule*.

5.3.4.1 Center Stowage Bin Structure Rules

The center substructure consists of an attachment bracket (*OHSCCYZBracket* to locate and fix the stowage bins two times in y-/z-direction) for the stowage bins at every second frame (valid for 4-frame and 2-frame modules). In the case of 3-frame stowage bins, the concepts foresees a bracket at every third frame. For 1-frame units at the end of the chain, both frames need a bracket (fig. 5.41).

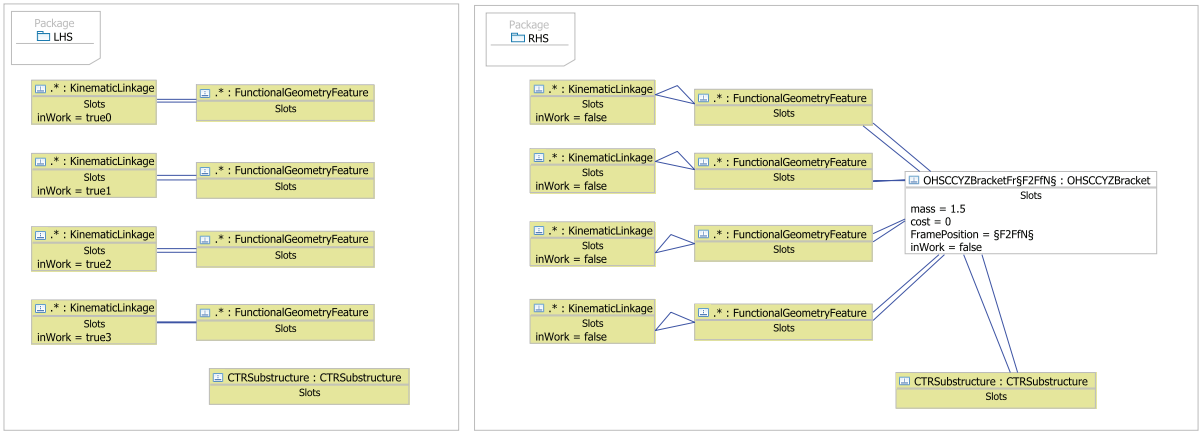

Figure 5.41: *Rule for inserting OHSCCYZBracket-instances into the model*

Any pair of subsequent *OHSCCYZBrackets* has to be interconnected with a plate (*CSSPlate, see rule shown in fig. 5.42*[111]). The first and the last center stowage bin are linked and fixed respectively to the first or the last *OHSCCYZBracket* in x-direction. Any other stowage bin in the chain is linked to the two adjacent stowage bins using an inter-bin connection plate.

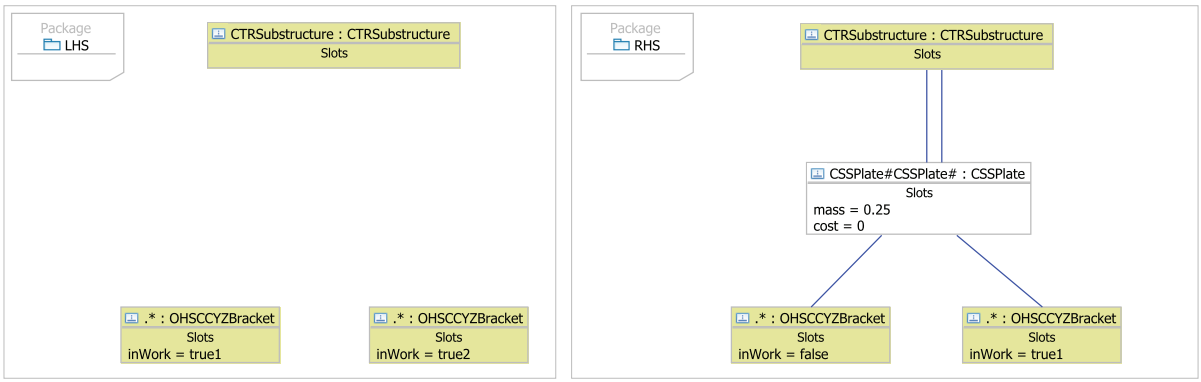

Figure 5.42: *Rule for inserting subcomponents of the central crown area substructure*

[111] The central substructure (here abbreviated by 'CSS') in the central crown area is used for the center stowage bin attachment. The 'CSS plates' mentioned form part of this substructure and are used to attach several system components including tubing, ducting and electrical wires.

5.3.4.2 Ceiling Panel and Air Grid Structure Rules

Ceiling panel and air grid attachment share one bracket at the junction of the two split lines. This bracket either has two *FunctionalGeometryFeatures* to attach the first or last pair in the chain (fig. 5.43), or it has four *FunctionalGeometryFeatures* to attach two pairs at any frame within the chain. The tolerance of the two or four features respectively is datumed back to 'LATSubstructureLH', and the features form an entity with a group tolerance.

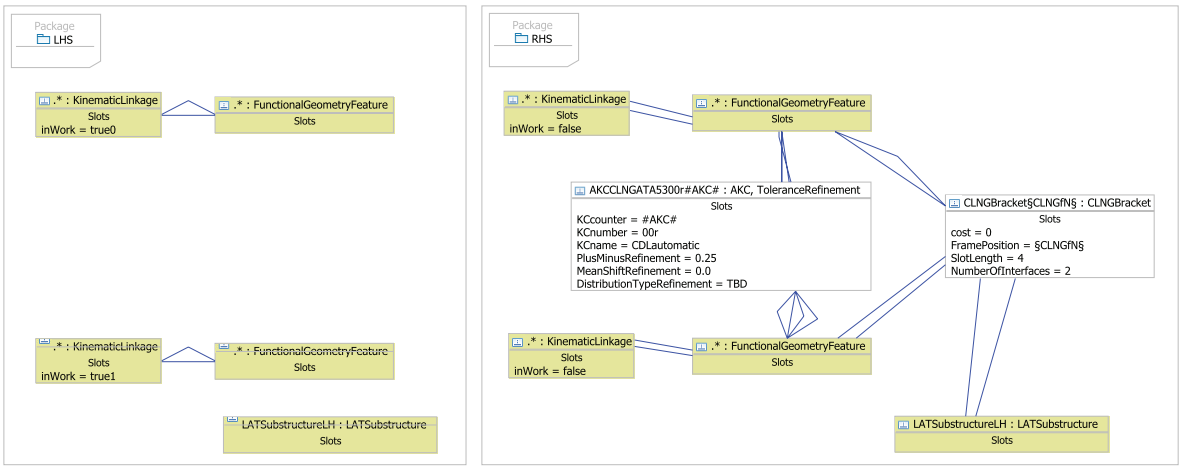

Figure 5.43: *Rule for inserting ceiling panel and air grid attachment bracket*

5.3.4.3 Lateral Stowage Bin Structure Rules

The attachment rules for the lateral stowage bins are comparable to those for the center stowage bins. The brackets for the lateral y-/z-attachment come every second frame for 4- and 2-frame modules, every third frame for 3-frame modules and every frame for 1-frame modules. The x-locating also follows the same principle as for the center bins. Additionally, the lower y-tie rod attachment has to be applied at every front or aft edge of the stowage bin. In the case of directly adjacent bins, there is only one y-tie rod. The implemented design rule sequence shown in figure 5.40 reads in detail as follows:

1. The program starts with a calculation all required variables.

2. A first loop iterates over the total number of frames within the investigated segment, which includes the cabin compartment module chain (*F2FchainFrameLength*) and the length of the implemented monument (*MonumentFrameLength*). Within the loop, one rule (red rule in fig. 5.40) links a new instance classified by *Frame*. The loop's control variable is assigned to its slot *FrameNumber*. This parameter contributes to an equation to calculate the explicit x-location of the frame within the global coordinate system.

3. The second loop iterates over *F2FchainFrameLength*.

   (a) The Java rule 'Activate OHSCCFrameParts' scans the model for center stowage bin instances which either start at, end at or cross the frame number as indexed with the current value of the control variable. The corresponding *MechanicalInterfaces* of this module are activated.

(b) Of the next four violet rules, a maximum of one rule will find the LHS topology pattern [134]. Either two center stowage bins meet at the current frame location, then the LHS of the rule 'OHSCC YZBracket Dual' will find a match and the corresponding RHS will be executed or one of the three others – 'OHSCC YZBracket FDW', if only a bracket *before* the first module is required, 'OHSCC YZBracket AFT', if only a bracket *after* the last module is required or 'OHSCC YZBracket middle', if the center stowage bin crosses the frame position and gets a thirds pair of interfaces with a load support function. Within each of these four rules, an instance of the type *OHSCCYZBracket* is created and linked to the *FunctionalGeometryFeatures* of the correspondingly activated *MechanicalInterface* instances. By default, the new instance is set inactive.

(c) The Java rule 'Activate CSSParts' checks if one of the instances classified by *OHSCCYZBracket* is marked (with the first iteration, this condition cannot be fulfilled). If none is marked, it marks the instance with the same frame position as the loop control variable. If it finds an marked instance, the slot *inWork* of the two instances is set to 'true1' and 'true2' respectively.

(d) On the LHS of the rule 'CSS Plates', two *OHSCCYZBracket*-instances with corresponding *inWork* slot values are searched. If found, the RHS inserts *CSSPlate*-objects representing the corresponding 3D visualization data and links it to the two corresponding *OHSCCYZBracket* instances. The length of these plates will later be calculated parametrically.

(e) The following Java rule 'Activate OHSCCParts 2' checks whether the current frame is the first or the last frame of the cabin compartment module chain, or if it is one frame in between. Accordingly, the neighboring stowage bin modules are activated.

(f) Comparable to the first four rules, the following three violet rules represent a selection list. Only the RHS of the rule, which has a LHS corresponding to the activated module(s), is executed. It inserts the *FunctionalGeometryFeatures* and the *MechanicalInterface* instances according to the position within the chain.

(g) The last rule deactivates the activated cabin module(s) and leads back to the loop's entrance described in step 3.

4. The third loop again iterates over *F2FchainFrameLength*.

   (a) The Java rule 'Activate CLNGParts' activates the *MechanicalInterface* objects at the frame location similarly to within the previous loop for the center stowage bin modules.

   (b) Either the rule 'CLNG Brackets Dual' or 'CLNG Brackets Single' inserts the *CLNGBracket* instance with the required parameters. In particular, the slot *NumberOfInterfaces* can be set to '2' or '4'. The slot has the UML stereotype *«CATIAparameter»* [16] and influences the geometrical appearance of the corresponding 3D data as well as the mass calculation for the bracket accordingly. Additionally to the bracket object, the rule links the involved *FunctionalGeometryFeatures*, using a tolerance refinement to establish the required group tolerance. The active *MechanicalInterfaces* are automatically deactivated, so the rule can immediately lead back to the decision node at the entrance of the loop described by step 4.

5. The implementation of the lateral stowage bin structure rules is similar to that of the center stowage bins. The only difference is that the four violet rules to insert the *OHSCLYZBracket*-instances ('OHSCL YZBracket Dual', 'OHSCL YZBracket FWD', 'OHSCL YZBracket middle', 'OHSCL YZBracket AFT') are followed by three violet rules for the *OHSCLYTieRod* instances, which are implemented accordingly.

6. The structure rules close with a final calculation of the design parameters.

5.3.5 Model Finalization and Analysis Transformations

After the execution of the aircraft structure rules, the CDL_LR design graph is fully developed (see fig. 5.44). The data model now is ready for the CDL model finalization and transformation rules. The second but last step of the root program (fig. 5.19) is a Java routine named 'CDL Model Finalization', which executes the validation routines within *cdl.validation*, the class operations within *cdl.operations* and finally once again the calculation of the design parameters. This corresponds to the phase *ensure model quality* (third phase according to section 4.4). At the end, the Java rule 'CDL Model Transformations' automatically executes several model transformation plugins within the phase *analyze model* (fourth phase) in order to generate the models for analysis and visualization purposes.

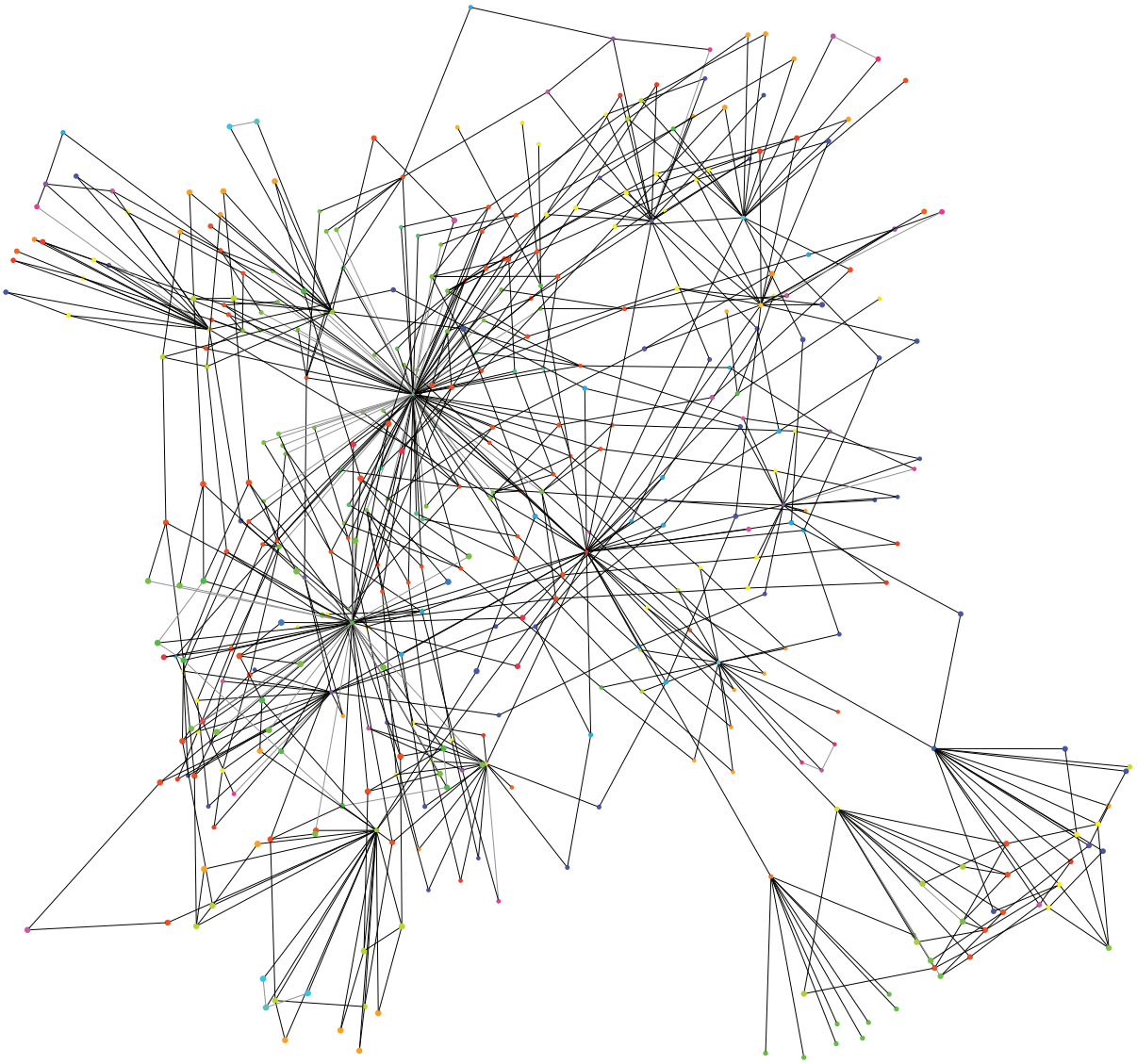

Figure 5.44: *Design graph of the baseline scenario after the execution of the aircraft structure rules indicating the size and complexity of the CDL_LR model (cabin compartment modules reduced to six frame bays to reduce data size). See fig. 5.15 for the corresponding DMU visualization.*

Chapter 6

Results and Discussion

6.1 Use Case Results

The step from a scientific engineering analysis to technical evaluation is an industrial work step and can only be outlined here. Within this section the analysis possibilities using the software interfaces introduced above shall be demonstrated and the industrial evaluation of the results can be sketched.

6.1.1 Trade Study 'Module Frame Architecture'

To analyze the multi-disciplinary repercussions of the different sets of cabin compartment frame architecture parameters, the 3D visualization (fig. 6.1) and the AAPs (fig. 6.2) can be consulted.

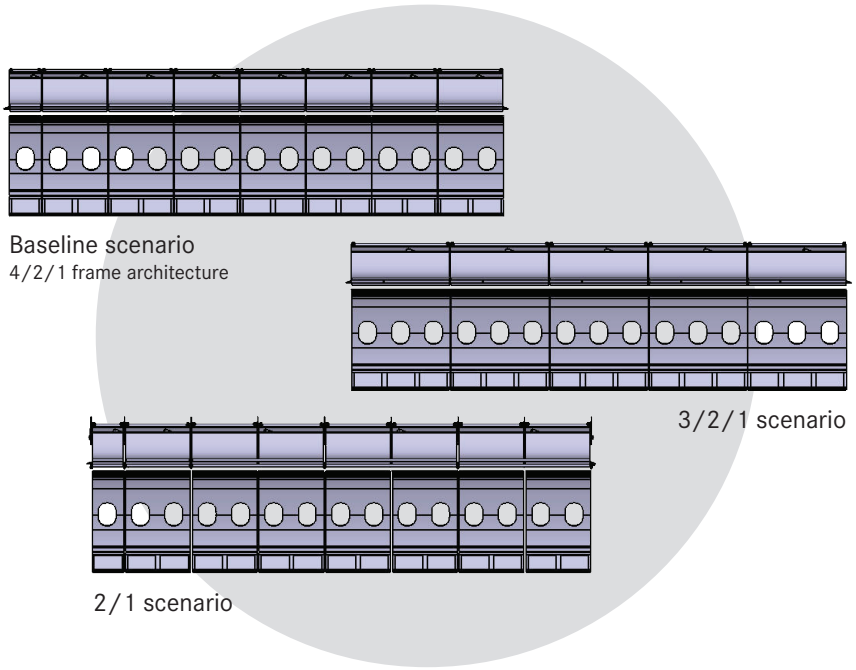

Figure 6.1: *Comparison of the 3D visualization indicating the different split line concepts of the module frame architecture scenarios. The 4-frame bins of the 4/2/1 scenario are not visible since they are hidden behind the 2-frame bin doors.*

Figure 6.2 on the next page shows some AAPs of the three scenarios mentioned: the 4/2/1 scenario, which is baseline, has the blue bars. The bars of the 3/2/1 scenario are colored dark red, and those of the 2/1 scenario are yellow. As can be seen, the 3/2/1 scenario is the most promising one concerning the cabin compartment complexity aspects. Especially the number of Mechanical Interfaces including the number of gaps or split lines is below the baseline scenario, which can also be anticipated in figure 6.1 showing the corresponding DMU. The number of cabin module and aircraft structure tolerances (MKCs and AKCs) is less for the 3/2/1 scenario, too. In both fields, the 2/1 scenario has the worst values.

The mass budgets for the cabin compartment modules and the attachment brackets lead to almost similar values. The relative mass increase for the brackets of the 2/1 scenario is associated with only small absolute values. The stowage volume as the chosen indicator for the cabin compartment performance shows limited differences for the three scenarios. The working time in the FAL seems to improve with the 3/2/1 scenario compared to the baseline scenario, while it increases for the 2/1 scenario. This goes back to a change of both the number of installation steps and a change of the number of cabin compartment modules, which need to be carried into the fuselage for installation.

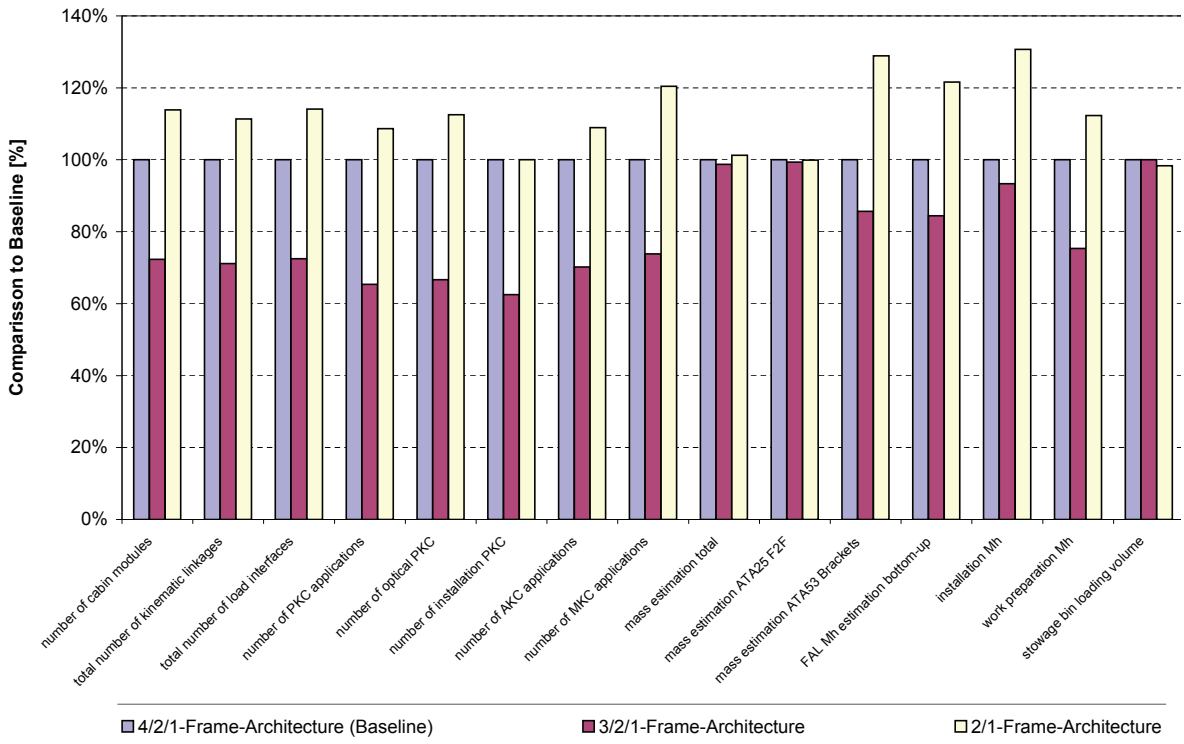

Figure 6.2: *Comparison of some AAPs of the module frame architecture scenarios*

After calculating and checking the AAPs, initial tolerance calculations are made to check for a consistent level of appearance quality and installation tolerances for comparable scenarios within the trade study family presented. While the rules for the synthesized MKC tolerances have been implemented into the model in the style of KBE data, the aircraft structure tolerance assumptions (AKCs) are implemented as fixed constant values only at this stage of modeling. Hence, before making final conclusion on the PKC calculation results, it should be crosschecked whether the aircraft structure tolerance assumptions match for the different substructure geometries, or if manual updated of the AKCs need to be made prior to making the final calculations.

This can be done by making use of the KC-list export feature provided by the plugin *cdl2kclist*. If an update of the AKC tolerances – or also of the MKC tolerances – is required, the data can be read back into the UML model using the re-import function of the same plugin. Then the plugin *cdl2mm* creates the updated MECAmaster input sheets and the PKC tolerances can be calculated by MECAmaster, followed by a detailed evaluation of the results (fig. 6.3).

– The calculations show a slight increase for PKC-002, which is the x-gap between two adjacent air grids. This increase's root cause is the increased airgrid manufacturing tolerances (MKCs) due to the larger panel size.

– Even though the larger ceiling panels face a similar increase in manufacturing tolerances, the final PKC-001 measuring the x-gap between two ceiling panels is reduced with the 3/2/1 scenario. The reason for this unexpected behavior lies in a reduced geometry factor of the ceiling attachment. The longer the panel gets, the smaller the rotational tolerances in x-direction are due to deviations in y-direction. Thus, the gap calculated with PKC-001 is 'more parallel' than the one of the other two scenarios – a good example of multi-domain repercussions of an architecture change being different than what has been expected.

– The y-gap tolerance between the ceiling panels and the adjacent air grids (PKC-003), the step in y-direction between two adjacent ceiling panels (PKC-007) and the installation tolerances at the pin-slot interfaces of the ceiling and air grid panel (PKC-100 and PKC-101) do not vary. The reason is that the input data including the parameters and the influencing topology have not changed for this aspect of the ceiling panels compared to the baseline.

– While the x-gap between the overhead stowage bins remains constant (PKC-004 and -005), the chain compensation tolerance seems to differ. However, the calculation of this particular tolerance in x-direction strongly depends on the number of contributors and on the resulting statistical distribution, as well as on the geometrical influence coefficient of the last stowage bin, which itself depends on its individual frame length. Especially the latter one differs between the three scenarios, resulting in the given differences. If the MKC tolerances can be controlled appropriately and can be kept constant for different module lengths, this effect can be compensated.

– The same applies to the gap between the last lateral overhead stowage bin and the monument (PKC-006), which is also influenced mostly by the varying geometrical influence coefficient of the stowage bin.

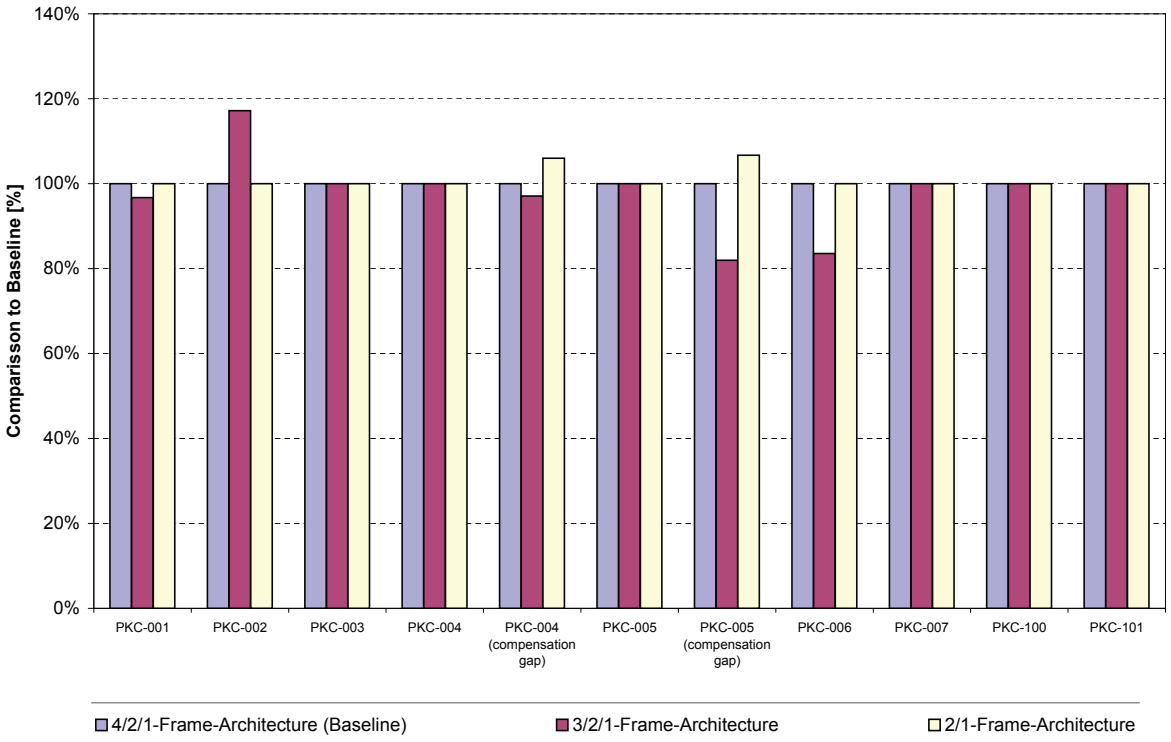

Figure 6.3: *Comparison of the PKCs of the module frame architecture scenarios*

In addition to the consideration of the AAPs, the MECAmaster linkage diagrams[112] are used to demonstrate the design complexity of the mechanical architecture aspects and can be the basis for technical discussions about the chosen linkage systems. On Moreover, the diagrams can be used to check if any model

[112]For explanations about the MECAmaster linkage diagrams also consider the corresponding notes at fig. 2.9 in subsection 2.3.2.

transformation error occurred or if a wrong input has been defined. Figure 6.4 provides an example for the MECAmaster linkage graphs for the baseline scenario.

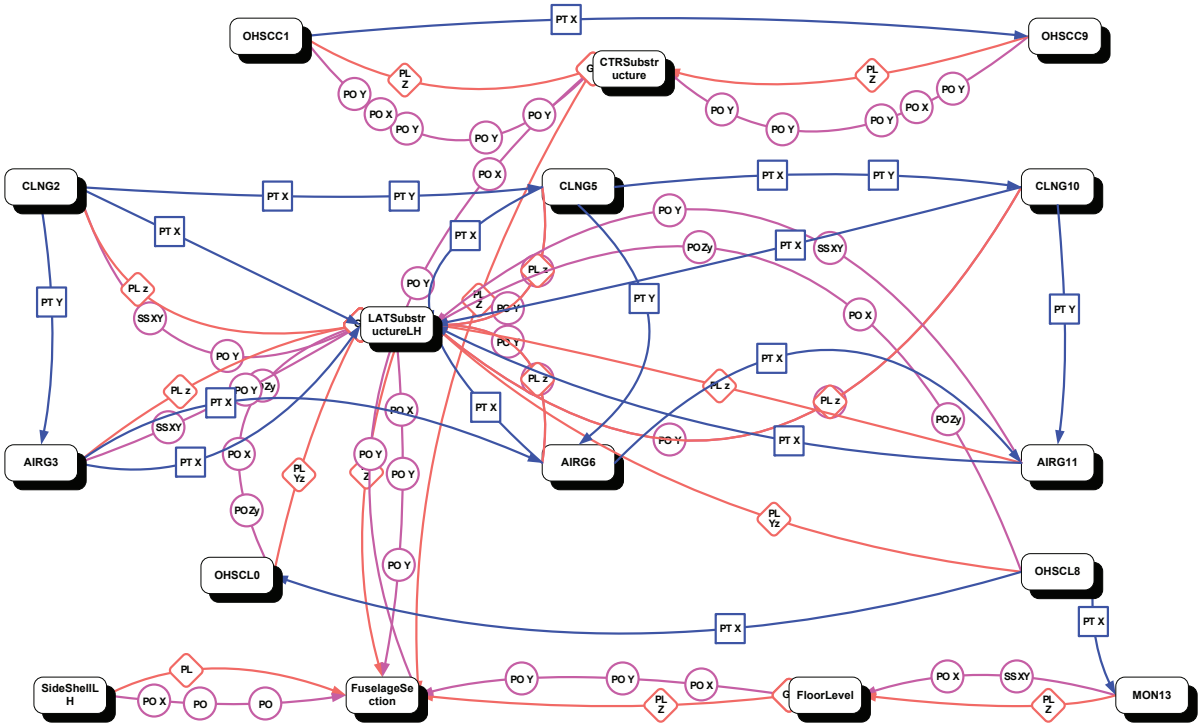

Figure 6.4: *MECAmaster linkage graph of the baseline scenario (extract of six frame bays) used for technical discussions about the linkage systems and for pragmatic modeling checks*

These formal facts compiled during the execution of the CDL can be interpreted so that for the consideration of cabin integration needs *alone*, the 3/2/1 scenario is the least complex one. For a new architecture and design this would mean that the complexity of the development work concerning mechanical cabin integration efforts can be reduced for the 3/2/1 scenario compared to the others with the given technical frame conditions. This conclusion is supported by a comparative look at the MECAmaster linkage graphs. These statements, however, neither pay respect to potential repercussions for the part number variety due to customization needs, nor can any conclusions be made concerning the different aircraft structure development efforts, as these aspects are not modeled or simulated.

Concerning the mass parameters, it needs to be recorded that altogether the deviation between the scenarios is of the same magnitude as the uncertainty of the chosen mass estimation methods and therefore no significant technical difference between the scenarios can be found. The same applies to the stowage volume estimation. This means that for two key aspects of the cabin compartment (mass and performance) no difference has to be expected between the three frame architecture scenarios. However, it needs to be pointed out that the stowage volume as chosen the indicator for the stowage space is perhaps not as precise as a concrete number of standardized bags or trolleys fitting into the stowage bins. A future task could be extending the CDL analysis methods with a more meaningful method to evaluate the stowage bin performance.

The handling of panels with increasing size may lead to additional work preparation time per module (see fig. 6.5, e.g.). When the installation time contributors have been modeled, this aspect has not been considered, but it may influence the overall required installation time. From a quality point of view,

the 3/2/1 scenario seems to lead to the best appearance tolerances. Additionally, neither the baseline scenario nor the alternative scenarios appear to lead to installation tolerance difficulties.

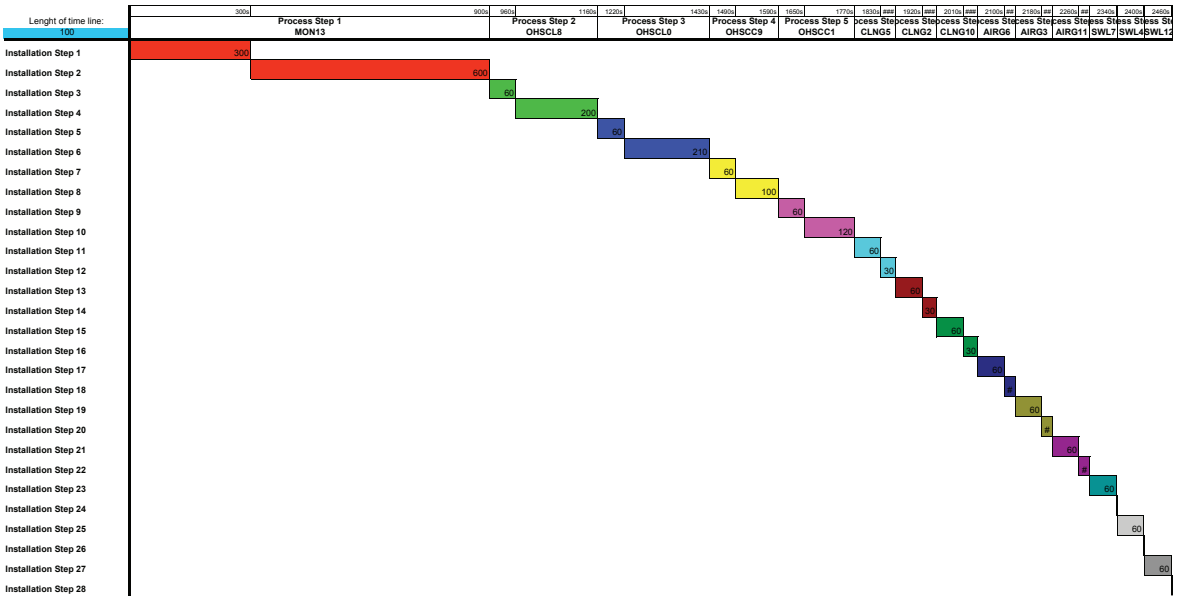

Figure 6.5: *Extract of the spreadsheet-based installation time and sequence visualization of the baseline scenario (extract of six frame bays) using the software interface cdl2fal*

6.1.2 Trade Study 'Frame Distance'

Altogether, 14 models with different frame lengths have been created varying between 400mm and 700mm (see fig. 6.6 on next page). In order to maintain a comparable length of the fuselage segment, frames have been added or removed compared to the baseline scenario with 530mm frame distance and 15 frame bays length. This leads to a variation of the overall segment length of $\pm 3,8\%$, which has to be considered for the evaluation and the comparison of the AAPs. Also here the AAPs are the first indicators for the differences between the 14 scenarios (see fig. 6.7 on next page). Even though there are some non-linear discontinuous effects between the smallest and the largest investigated scenario, only the extreme scenarios are shown to simplify the visualization.

The different frame distances lead to huge differences concerning the quantitative design complexity, as the corresponding AAPs tell: the smaller the frame distance, the more interfaces and all correlated elements such as brackets, tolerances to be controlled and installation steps. The qualitative design complexity, however, can be considered to remain at the same level, as only the number of features increases, while the interdependencies are generically the same on class level.

The relative mass of the cabin modules increases for larger frame distances, as both the panels as well as the stowage bins need reinforcement applications to withstand the increasing bending and torsion loads. The mass of all cabin attachment brackets decreases due to the fact that larger frame distances lead to fewer interface points and thus to fewer brackets. An 'optimum' for the individual parts together with the brackets cannot be simulated here, as the mass calculation formulas are rough estimations only. In any case, the tendency becomes clear. Still, aircraft structure masses need to be considered in parallel to make final conclusive statements beyond a pure optimization of the cabin mass.

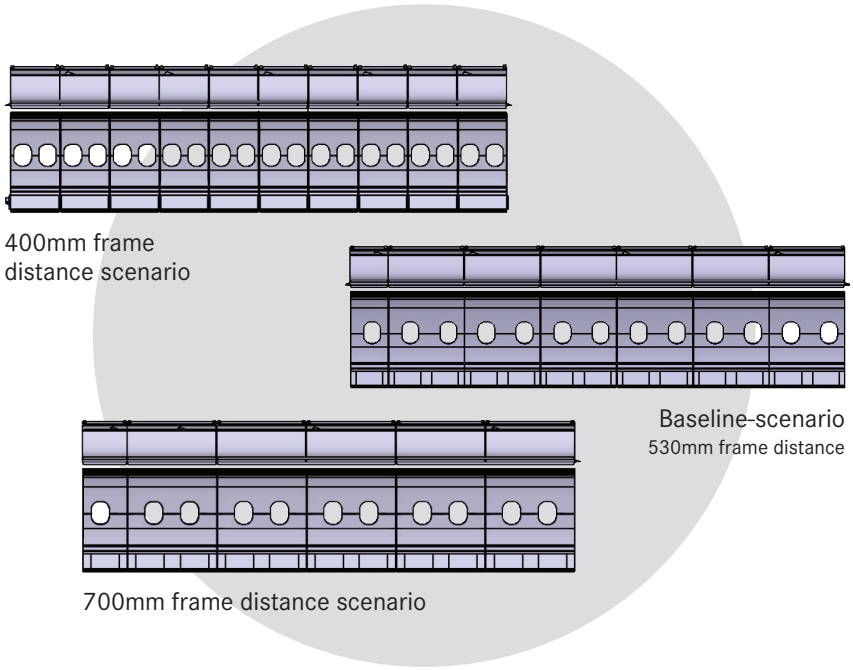

Figure 6.6: *Comparison of the 3D visualization indicating the different frame distances of the frame distance scenarios*

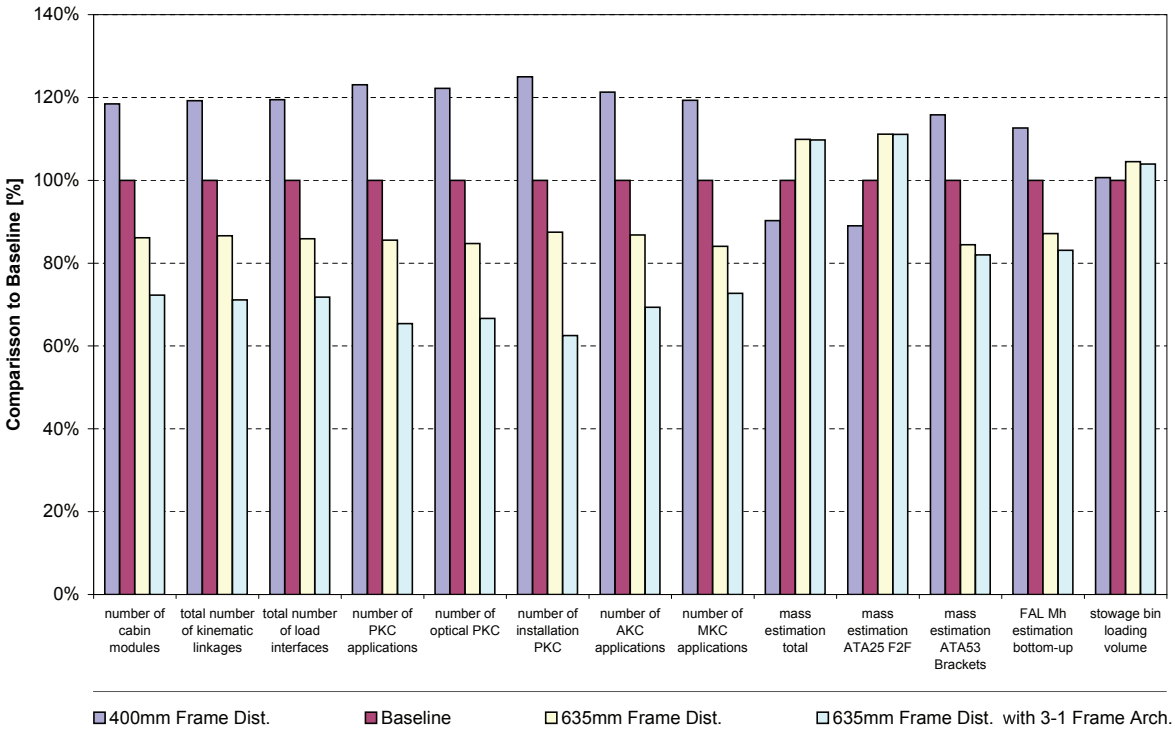

Figure 6.7: *Comparison of some AAPs of the frame distance scenarios*

As already mentioned, the installation time goes down with a reduced number of mechanical interfaces, but the repercussions of larger modules on the handling efforts may absorb a bit of the anticipated installation time gain. Both the quality-related PKCs as well as the installation PKCs show no significant change within the investigated frame length interval. Beyond the 700mm frame bay, however, increasing installation and quality issues need to be expected, if comparable tolerance behavior of the modules and of the aircraft structure is premised.

A conclusive qualitative statement for the frame distance scenarios is that varying the frame length means playing with the quantitative design complexity versus the mass. When searching for a 'local optimum' (cabin-only) or even a 'global optimum' (cabin and fuselage) for these two aspects of mechanical cabin compartment architectures, one should consider the potential benefits of combining the variation of the frame length and module architecture. Figure 6.7 comprises a combined scenario which supports this assumption.

6.1.3 Trade Study 'Repercussions of a 25mm Gap'

The iterative trade study about the repercussions of a 25mm gap starts with a change of the architecture parameter 'NominalGapUnloaded2Unloaded' from 10mm of the baseline scenario to 25mm. The first results from the AAPs are that the mass of the cabin compartment modules is reduced slightly due to the larger gap. The stowage space of the overhead bins is reduced in parallel. Using the adapted 3D geometry generated automatically by the software interface to CATIA V5, DMU investigations can take place to check if the gained space between the stowage bins can contribute to significant installation handling simplifications and therefore to a potential reduction of the corresponding installation step time.

Additionally, it should be investigated if the spatial changes offer to locate functional applications in the gaps such as system or wire routings, which could improve the design at other locations. By the aid of the automated tolerance calculation preparation (fig. 6.8) it can be demonstrated that the relative appearance PKC tolerances[113] improve (see fig. 6.9 on the next page). The installation PKC tolerances do not face any impact. Thereby it is assumed that the MKC and AKC tolerances remain at the same level. Globally seen, neither recurring nor non-recurring cost repercussions have to be expected with this scenario alongside the mentioned loss of stowage space and the light cabin mass reduction.

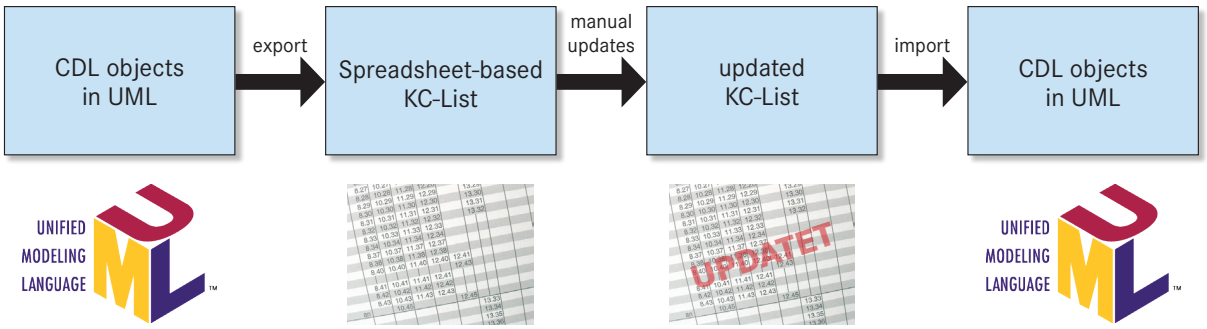

Figure 6.8: *Round trip process for AKC and MKC tolerance updates*

Based on the assumption that increasing tolerance limits can reduce manufacturing cost, an iteration loop to the first scenario is performed (fig. 6.8). Now the aim is to increase the AKC and MKC tolerance limits of the 25mm gap scenario such that the relative PKC tolerances are still equal or better than the original baseline values, and also that a manufacturing cost reduction is tangible.

[113] Relative tolerances are the PKC tolerance values divided by the corresponding nominal gap size.

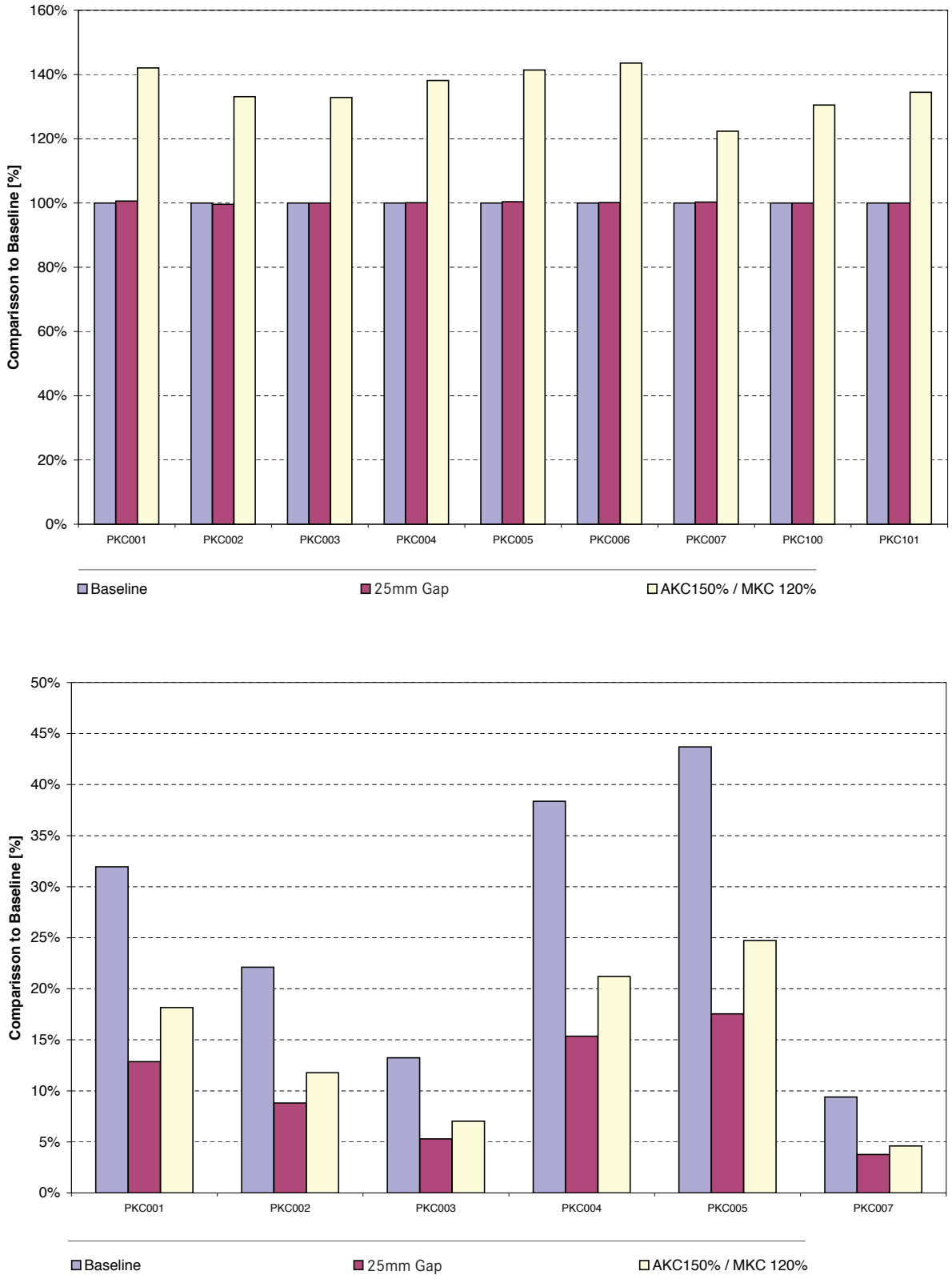

Figure 6.9: *Comparison of PKCs of the 25mm gap scenarios: while absolute PKC tolerances increase, the relative PKC tolerances for a 25mm gap decrease even with increased AKCs and MKCs (AKCs $+50\%$, MKCs $+20\%$).*

During discussion loops with the corresponding manufacturing and tolerancing experts for the cabin modules and the aircraft structure, a tolerance increase of 50% for both the AKCs and the MKCs has been judged as significant for cost improvement. The changed tolerance values can now be read back into the CDL_LR UML model by using the round trip function of the plugin *cdl2kclist* (fig. 6.8), followed by a re-compilation of the MECAmaster input file. The calculations showed that the relative PKC tolerances are still below the baseline (fig. 6.9). Nevertheless, PKC-100 and PKC-101 representing installation tolerances are also impacted: the uplong holes for the ceiling panel and air grid attachment are no longer large enough to compensate for the tolerances and need to be enlarged by approximately 50%. The corresponding brackets now face a size increase and therefore the mass of the brackets increases as well.

The size increase is incorporated into the CDL_LR model in the corresponding rules by changing a design parameter of the associated *SubComponent* instances. After re-executing the entire CDL_LR rule set, a corresponding mass increase evolves. Neither the stowage space nor any relevant spatial aspects are affected.

The conclusion for this iterative study is that a 25mm gap between the cabin modules could contribute to an improved tolerance-related appearance and even to a small but remarkable weight reduction along with the potential to cut back some tolerance-related recurring manufacturing cost. The price for these improvements is the loss of some stowage volume and larger nominal gaps which may create general appearance problems.

6.1.4 Further Possible Scenarios

Besides the discussed scenarios, further scenarios based on the present CDL_LR model could be:

- **Any combination of the described scenarios** The semantic hull built by the case study model in conjunction with the generic CDL class model can be used to create further consistent scenarios using the same model data. Due to the fact that the design rules used are generalized and therefore are independent from the input parameters, it is possible to overlay any combination of the previously described scenarios directly without any further modeling efforts by choosing the proper input parameter combination.

- **Modification or replacement of an entire cabin module** It is possible to exchange an entire cabin module by a modified or an all-new cabin module with a different topological composition, for instance with alternative attachment brackets including different interface functions and locations or with different geometrical shape with different split lines and gaps. The latter can comprise trade studies for different industrial design contours and the analysis of the corresponding repercussions. The required modeling efforts include exchanging the encapsulated design rules for the cabin module with new rules and parameters and adapting the corresponding substructure rules.

- **Implementation of additional cabin modules and system components** Omitted cabin compartment modules like the sidewall panels and the dado panels, door frame and entrance area lining panels or even monuments and crew rest compartments could be implemented as well as cabin and aircraft system components like air distribution ducts or cabin data system computers. The corresponding *physical* integration aspects can be considered in the same way as those of the cabin modules. Besides some additions to the architecture rules, it would be required to extend the cabin module and aircraft structure rules accordingly.

- **Change or extension of the current structure integration datums** If the allocation of mechanical interfaces to structure components and of AKCs to their respective integration datums change (for example, if additional integration datums are incorporated) it would be necessary to adapt the substructure rules so that the mechanical interfaces and the AKCs are linked to the correct objects.

– **Implementation of a cross section parameterization** It would be necessary to model all parametrical and topological repercussions in the cabin modules split lines and structure components. In particular the complex architecture differences between a single aisle and a double aisle aircraft (the first one without center stowage bins and therefore without the center substructure) require several additional architecture rules. The gap rules, the cabin module rules and the aircraft structure rules, however, would only need extensions or modifications like the interchange of 3D data or an upgrade of the geometry and tolerance parametrization.

6.2 Methodological Results

6.2.1 The Implementation with Design Languages

The CDL and the subsequent implementation process as presented in this work are not 'artificial intelligence' approaches [48, 131] and cannot randomly create new design proposals. It is and remains the task of the involved engineers to develop concepts or images of cabin architectures and then use modeling techniques such as design languages to capture and analyze the concepts with the aid of software-implemented data models.

On top of representing the scenario-related classes within the class diagram, the chosen approach with design languages additionally offers an automated design processing which helps the engineers to focus on the development of concepts and on the collection of data rather than on complex design compilation, affirming the second hypothesis from chapter 3 concerning the *model generation aspects*.

The selected working process resembling a tolerance management process (see section 4.4) implicitly guides the modeling engineers to the question of 'which data is required' in order to provide answers to the analysis questions, which proves the first hypothesis from chapter 3 valid. The work processing leads to the necessity to cluster the design rules. Such clustering, however, implies that repetitive and iterative calculations need to be performed.

In the CDL_LR model, equations need to be solved several times and some calculations are even decoupled. In fact, this inhibits the reversibility of these calculations in particular, but due to the definition of a certain rule sequence as modeling philosophy these reverse calculations are not required anyway. Within the chosen implementation, the modeling sequence of the four main phases cannot be interchanged[114], but subprograms such as for instance the compilation rules of a single cabin module can be exchanged in a modular way.

The fourth phase of modeling according to section 4.4 consists of model transformations for analysis purposes, which are conducted by the plugins introduced. In this respect, the plugin *uml2mecamster* can be considered as more or less 'complete' and ready for wider testing. All MECAmaster model elements find their corresponding routines within the plugin. Any model which can be created with the MECAmaster GUI can be created with a graph-based design language using MECAmaster vocabulary, too.

Due to the similarity of the required input data, it is possible to exchange the kinematic tool MECAmaster by other kinematic CAT software or even with Monte Carlo simulation-based software, such as for instance 3DCS. This independence from commercial analysis models ensures flexibility for potential software changes and stands for the implementation-independent definition of the CDL class diagram. Since both kinematic tolerance analysis and tolerance simulation using Monte Carlo algorithms require similar data input, the decision to go with one specific CAT tool family does not have to be considered as a one-way solution, but leaves the door open for later software implementation changes or extensions.

[114] It has to be noted that the sequence of a 'manual' compilation and execution of such multi-domain trade studies including the creation of tolerancing models and tolerance calculations – which also had to follow the descriptions of fig. 4.10 from section 4.4 – could not be interchanged, neither.

The CDL-related plugins have to be considered as a test case. At this point in time, application experience needs to be collected. The embedded range of functionality for class-related operations within the plugin *cdl* fulfills the required spectrum which has been described by the *scope of modeling* (section 4.1). Of course, the operations can be adapted at any time, if the scope of modeling needs to be extended. The working of the analysis plugin *cdl2analysis* has been very helpful. Also here, code changes are required only if further AAPs are defined and need to be calculated in an automatized way.

The model-to-model transformation routines of the plugin *cdl2mm* currently only 'know' three different linkage patterns [134] for the cabin modules plus the linkage systems for structure components. For these, the conversion works properly according to the tests. If an extension is needed to further linkage systems, like those for cabin monuments or for detailed structure modeling, additional patterns and the corresponding transformation rules need to be programmed. Especially the transformations from CDL UML vocabulary to MECAmaster UML vocabulary are a good example for complex model transformations with a high level of abstraction, where design languages using a model-based paradigm unfold their full strength compared to pure text-to-text transformations. Without this high level of abstraction, such generalized transformations are difficult to realize.

The plugin *cdl2knlist* fulfills all needs for a round trip plugin between the CDL model in UML and the spreadsheet-based KC-lists. As already mentioned, topological changes should not be managed by such model-to-text and text-to-model round trips, but by the rule-based model generation itself in order to ensure repeatability, automated calculation of multi-domain repercussions and data consistency. The FAL plugin *cdl2fal* is exemplary. If using this plugin and its automated visualization scheme with spreadsheet-based bar diagrams works out, it is possible to extend it to parallelized installation steps. Due to the modular data structure, it is also easy to exchange this plugin with an interface to more professional process visualization models or even to digital factory models [10].

The export plugin *cdl2export* only aims at demonstrating the flexible possibilities to exchange any modeled CDL data with any target application. For transformations to product databases or PDM systems for industrial work, similar transformation algorithms can be applied, but case-specific model validations need to be performed, which cannot be implemented in a generic way.

Altogether, the plugins can fulfill the requested tasks and thus verify the model analysis aspects of the second hypothesis from chapter 3. Even though there is no primary focus on the GUI, but more on the running model implementation, the resulting user interface performs in the way which is needed for conceptual studies. The data import of the input architecture parameters for the axiom within the plugin *cdl* has been realized with a spreadsheet-based interface, which, however, is fully in line with industrial practice. The same applies for the output visualization of the analysis plugin *cdl2analysis*, of the FAL plugin *cdl2fal* and of the KC-list plugin *cdl2kclist*. Especially the last one matches directly with the industrial practice of maintaining KC-lists within spreadsheets. It is only for the transformation plugins *cdl2mm* and *cdl2export* that simple GUIs have been developed, because here user interaction is mandatory for the selection processes.

Depending on the concrete modeling need, the user can choose between graphical rule implementations or Java-based coded rules. In particular for the creation of new data objects and for the search of simple topology patterns [134], the graphical rules were revealed to be a fast and pragmatic procedure. The graphical rules only come to a limit if complex object patterns need to be searched and if a mixed topology and parameter search is required. In such cases, it proved to be helpful to mark single objects as 'objects in work' with the help of Java rules[115]. In a following graphical rule, this marked object can now be processed in a simple way. The necessity to be able to retrieve instances and instance patterns has to be considered when the corresponding instances are created. This means that rule-based modeling sometimes is not a linear creation process, but rather an iterative process.

[115]The CDL implementation foresees the slot *inWork* to mark individual instances, as has been explained in chapter 5.

Fully generic rules, which do not use CDL_LR classes but only generic CDL classes could easily generate and extend topology. But such generic rules are weak for linking specific instances with specific other instances and for adapting the design parameters to specific values. So the chosen concept of a case-specific class diagram – the CDL_LR class diagram of the presented case study – is a helpful method.

The implemented CDL_LR design rules follow a modeling philosophy of a balanced mixture between generic and individualized rules as a pragmatic approach. The case-specific classifiers are taken for the model creation – the more abstract CDL classifiers which the case-specific classes inherit are considered for model analysis and generic model transformations within the model-to-model transformation plug-ins (*cdl2*x). It is interesting to note that the case-specific class diagram together with case-specific rules could also be interpreted as a rule-based knowledge representation in the sense of KBE databases.

6.2.2 Links to Further Analysis Models

The chosen scope of modeling has been selected as a pragmatic approach to provide answers to questions in the context of physical aircraft cabin architectures. Of course, a larger scope could have been chosen, implementing further disciplines with their methods. One upper limit surely comes up, if one single engineer cannot control the entire span of involved methods and models. This aspect and related issues are discussed within section 6.3. Some analysis methods 'in the methodological neighborhood' of this cabin architecture analysis approach are left out for pragmatic reasons. While FEM-based load and stress analysis is required to shape physical components and to calculate the acting loads, it would be overdone to implement these analyses already during the working loops conducted with this methods.

The same applies for the question of whether to model aspects related to total cost and life cycle cost. The execution of load and stress analyses or extended cost modeling can be considered after the creation of initial CDL models. Therefore, the generated data models can be used as a basis together with the provided export plugin *cdl2export*. For the studies like the given CDL_LR trade study, a link to functional system models is not required at this stage, because the interrelations between cabin modules and system components mostly comprise physical aspects only – at least on an architecture level.

Having the chosen chosen scope of modeling and its purpose in mind, the analysis models, which are outlined subsequently are interesting candidates for further software interconnections beyond the initial purpose [155] of CDL models.

6.2.2.1 FEM-Based Load Analysis

Using the definition of Mechanical Interface objects, the CDL model already contains much of the relevant data needed for load analyses with FEM. Together the center of gravity and the mass, the locations and the functional descriptions of the *LoadInterfaces* provide boundary conditions required for FEM models. The missing information concerns a suitable meshing of the geometry data including a model to describe the material behavior and the definition of load cases, which will impact the cabin module.

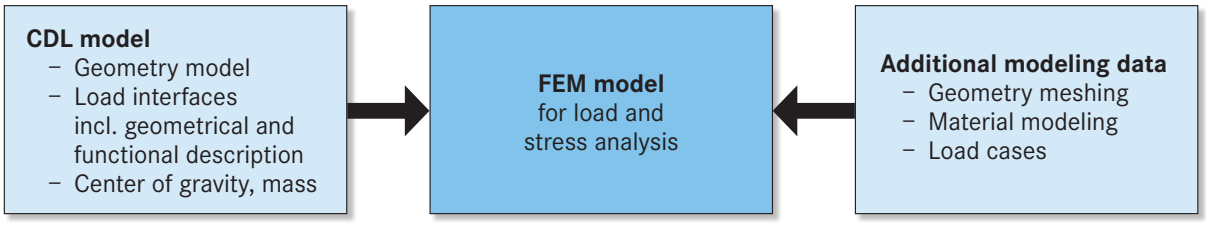

Figure 6.10: *Data transfer from a CDL model to a FEM model*

Meshing data is considered as more detailed design data. It is not implemented in the CDL model directly and is also outside the scope of this work. As mentioned, load-stability is taken for granted for the investigated cabin modules. If FEM analyses are planned after the execution of CDL analysis, it is possible to append meshing data to *CabinModule* classes in the same way as it is done for geometry data. After that, FEM simulation input data can be compiled directly. Figure 6.10 sketches the potential data transfer from a CDL model to a FEM model, where the dark blue arrow represents a potential standardized model-to-model transformation. Experience has already been gained incorporating meshing data into design processes using design languages [64].

The software link between design languages and FEM software already exists [85, 86, 87]. KORMEIER demonstrates the flexible functionality for topology synthesis [86] and material modeling [87] and shows first examples for embedded FEM analysis in conceptual design processes supported by design languages [64]. It is possible to think of comparable rule sets and algorithms for the synthesis of the inner topology and of the material models for Cabin Module FEM studies. For instance, knowledge-based automatized meshing algorithms could be deployed [64].

6.2.2.2 Functional Cabin and Systems Integration

It is possible to think of architecture concepts for cabin modules and systems, which are different from the CDL_LR example and where *functionalities* of aircraft systems are closely coupled with mechanical cabin module integration. For instance, cabin modules could contain integrated system subcomponents, where the system behavior directly influences cabin module design parameters and therefore the cabin module architecture.

For such cases, it is possible to create system-specific classes, which inherit from the CDL class diagram and which are extended with additional system-specific classes[116]. This could for instance be classes representing physical objects like wires, connectors, ducts or hoses, but of course also classes for functional system modeling linked to corresponding analysis or simulation software [107, 128].

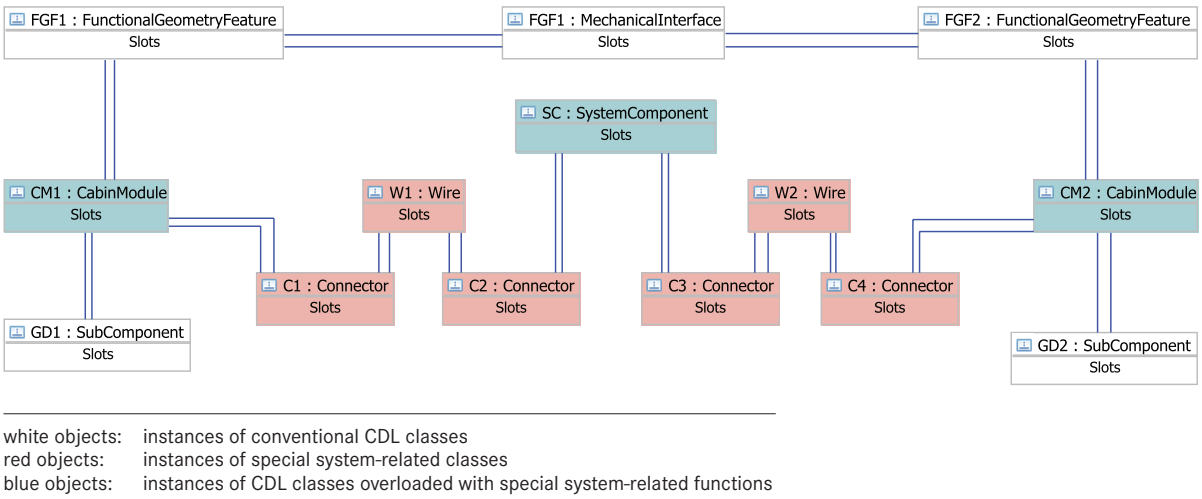

Figure 6.11: *Design graph with CDL objects and special systems-related objects*

The simple example given with figure 6.11 shows that looking at physical aspects of cabin architectures does not fade out systems-related, functions-related or operations-related architecture aspects. In spite

[116] The overlaying of UML instances with multiple UML classifiers is common practice for multi-domain modeling.

of the chosen CDL analysis methods for physical aspects of ATA25 modules (section 4.4) separate from cabin systems and functions analyses, the need for joint evaluation or even model coupling as expressed with the first hypothesis from chapter 3 can be fulfilled.

6.2.2.3 Link to Digital Factory Models

Based on the work of ARNOLD [10], it is possible to link design languages in general – and therefore also CDL models in particular – to digital factory models. The installation sequence of the modules and all installations steps including the sequence already form part of the CDL class definitions and are used for the visualizations based on the plugin *cdl2fal*.

Comparable to the addition of systems-related data, additionally required assembly data can be added to the CDL objects [130]. Therefore, the CDL objects can be overloaded within the scenario-specific class diagrams. A transformation plugin would be needed to add the missing information using databases or by user input.

6.2.2.4 Links to Databases, KBE Systems or PDM Systems

In the example of the CDL_LR model, some spreadsheet-based input data has been imported. This principle of data base implementation can be extended to larger data base applications and even to the extent of knowledge-based input. This knowledge-representation can either be realized with libraries for graph-based or Java-based rules or even with complex design patterns [134] or high-level primitives [92].

The transition between a data model for analysis purposes and a multi-domain product data model for PDM and PLM purposes can be considered as smooth concerning data transformation and handling. The geometry data within CDL data models, for instance, can be resorted in a rule-based manner to match PDM structures [16], and the same applies to any other structured data object domain. To fill the aforementioned gap between manufacturing cost and total cost or life cycle cost respectively, further analytical decomposition and synthesis approaches would have to be linked with the CDL model.

It is possible to think of overlaying classifiers similar to the potential embedding of systems analysis and simulation data. Of course, here is the possibility for a close link to the aforementioned digital factory data model to estimate the total cost. A mapping of executable product requirements on the design model in the style of RBE methods is currently not implemented explicitly. Due to the CDL and UML model flexibility, it is possible to extend the current modeling domains by requirement data and a link to RBE methods including the corresponding data models is possible.

It should be mentioned that data round trip processes between the CDL model and a PDM system are possible, as long as the model-to-model transformations only extract data without changing the data object structure. The demonstrated plugin *cdl2export* can handle the export functions in a generic way. However, the transformation backwards from PDM to the CDL models becomes very sophisticated as soon as structural or topological modifications have taken place inside the PDM.

6.3 Way of Working in Industrial Context

In the industrial context, any new method needs to be proven to be applicable. Even though in theory methods should drive the working processes, in reality industrial frame conditions slow down or even block the introduction of new methods. In other words: sometimes, processes drive methods.

Rule-based design languages offer to ‘design’ the methodological sequences flexibly for such problems [90, 134]. Subsequently, some requirements for a pragmatic industrial implementation of the proposed cabin design, architecture and integration analysis method using the CDL are outlined, making use of this advantage.

6.3.1 Role of Cabin Architect and Expert Group

For CDL-based analyses, the role of cabin architects is integrating the multi-disciplinary results and attempting a holistic evaluation of the analysis data.

- The cabin architects need to be familiar with the full bandwidth of the involved disciplines including their methods. This comprises technical understanding about the analysis models evolving from the fifth phase of the working process according to figures 2.13 or 4.10 respectively.
- Together with a data management engineer (see following subsection), knowledge is required about the limits of the CDL model. This comprises the knowledge about the data depth, about the simplifications and assumptions on a technical level based on semi-analytical, empirical or even heuristic methods.
- Additionally, particular knowledge about the scope of modeling is mandatory for robust and valid model results.
- Generally speaking, design language-based iterative trade studies can comprise two levels: the iterative recompilation of the design data in order to find a suitable input value to reach a desired output value, as well as the iterative extension of the class diagram and of rules – and therefore the iterative extension of the semantic hull or even of the scope of modeling in order to improve the pragmatic sense of the design language model[117].
- The CDL models and the corresponding AAPs should be considered as indicators of tendencies and not as autonomous or holistic evaluation statements – especially not in the sense of the pi-theorem [133]. The same applies to any other model result parameter like PKC tolerance results or installation time estimations. The model's purpose is to support the cabin architect with his or her judgment and to prove heuristics with profound model results, not to replace intuition and technical expertise.
- In addition, knowledge is required about non-formal product data such as industrial design or cabin customization aspects, which cannot be modeled explicitly. Aspects which are not in the model, but are important anyway should be considered 'in the old way'. However, with the presented method it does not require much time to reproduce various trades with different industrial design geometries or with alternative attachment concepts to light implications for customization and industrialization.
- The final holistic evaluation about a technical cabin integration scenario is not 'the sum of all analysis parameters', but a holistic consideration of the modeled formal results along with the omissions of formal and non-formal nature. It needs to be clear that in order to know *how* to evaluate, it is necessary to know *what* to evaluate. A frequent 'hermeneutic dilemma' for this point is that in order to know *what* to evaluate, it is necessary to have an idea about *how* to do it. This means that as cabin architect it is mandatory to challenge both the *technical results* on instance level *and* the class diagram with the corresponding analysis criteria representing the *scope of modeling*.

[117] For example, when the installation PKCs of this scenario had been calculated using the new tolerance values, the results showed the need to increase the sliding features of the brackets to compensate for the increased installation tolerances. Therefore, the mass calculation formula for the brackets has been extended by a parameter for the length of the sliding features as a second iterative step. The original value leads to the bracket mass as it has been used in all previous scenarios; hence the consistency of this model with the previous ones is guaranteed.

Where needed, the cabin architect can consult an expert group supporting the modeling and analysis phases (fig. 6.12):

- An engineering expert for cabin-related interface and tolerancing issues like PKC definition, MKC capabilities, interface functions, etc.
- An engineering expert for cabin module design and manufacturing including FAL processes responsible for the design data of the cabin modules and of the attachment brackets
- An engineering expert for aircraft structure-related interface, tolerancing and manufacturing issues like AKC capabilities, brackets, structure manufacturing process possibilities etc.
- A engineering expert for cabin, bracket and aircraft structure stress and loads aspects
- A data management engineer [93] who is responsible for implementing the modeling and the model transformation methods including the software interfaces to any required analysis methods

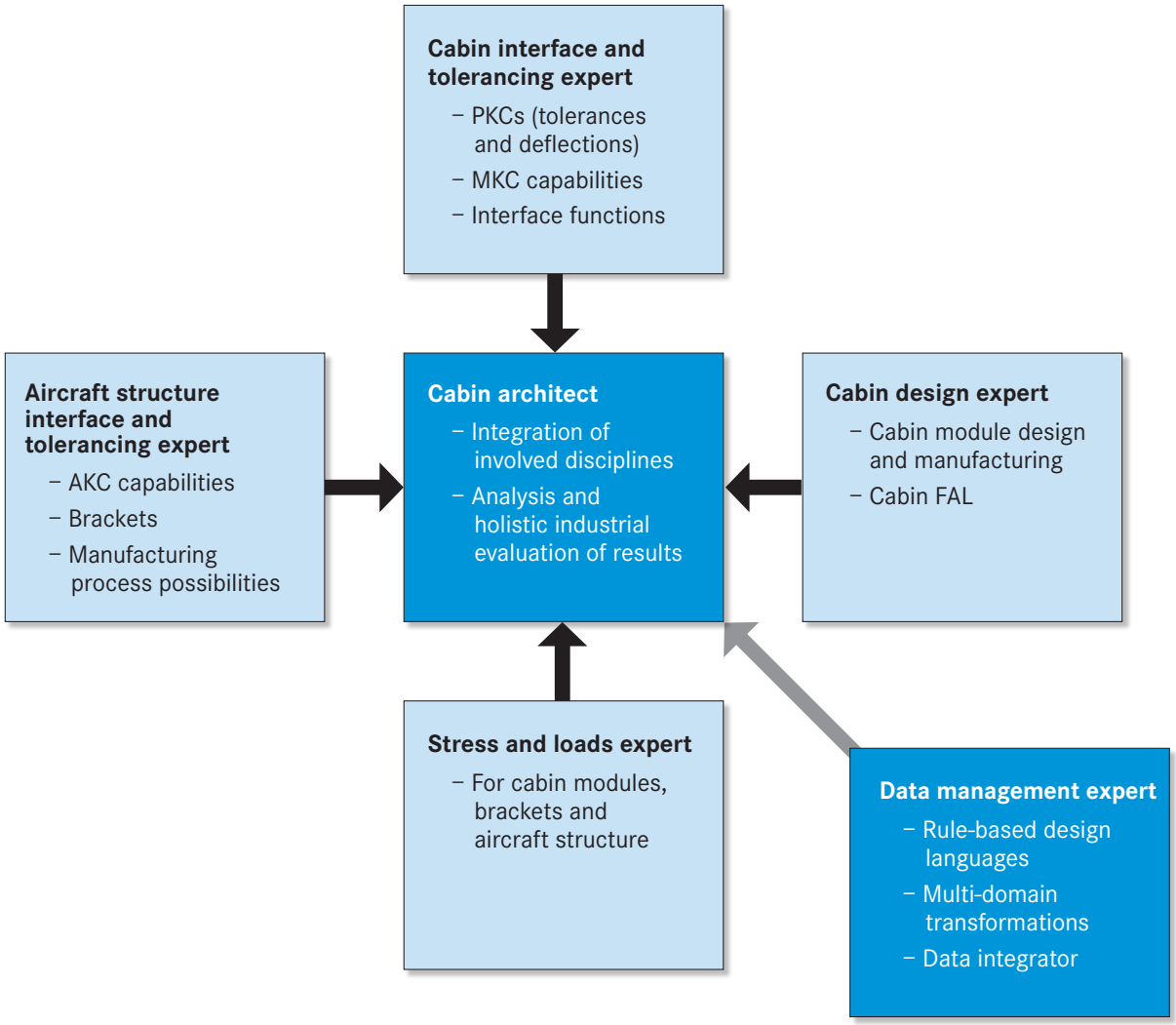

Figure 6.12: *Industrial working group for the cabin architecture analysis tasks*

6.3.2 Industrial Challenges

Based on the experience gained with this research work, it can be said, that three types of users can be distinguished with different levels of knowledge about design languages:

- Simple design language execution and parameter changes require a basic know-how about the involved class diagram respectively the vocabulary along with the existing rules and model transformations. Basic training can enable design engineers to perform such tasks.

- Advanced parameter trade studies with a modular exchange of rules and objects require advanced user experience with associated training needs.

- Adaptations to the generic CDL class diagram, the scope of modeling or the model-to-model transformations for analysis and visualization purposes require expert knowledge about design languages, the model-based paradigm and about the specific engineering disciplines. Aside deeper trainings, working experience with design languages in the range of several months is a prerequisite for such modifications.

In order to enable multi-disciplinary work, a common standardization of vocabulary [136] such as the definition of the term 'mechanical interface' is required, including the corresponding overlapping of the different conceptual ideas of these classes. The proposed functional differentiation from chapter 4 can be seen as examples fulfilling this need. It is necessary that the people involved in the working group understand the core principle of CDL with the rule-based paradigm and with model-to-model transformations for data analyses. This comprises adequate training and first working experience with design languages in order to be able to execute design language models. Additionally, chances and limitations of design languages using a model-based paradigm need to be communicated in general to create awareness for changes and for a pragmatic and fruitful use of design languages:

- Design languages are no kind of 'magic artificial intelligence'[118].

- Only formally understood problems can be modeled exactly.

- Semi-analytical or empirical aspects of models need to be outlined.

- In particular, 'fuzzy' aspects like industrial design may be crucial for cabin architecture decisions, but are difficult to be modeled and analyzed in an explicit manner.

- It needs to be made clear to any user or any involved person providing expertise that models can only answer to dedicated questions. No phenomenon which has not been modeled will 'evolve' or 'emerge' out of the model.

The method proposed using CDL focuses on a restricted purpose, which is conceptual architecture analysis. It lies in the nature of design languages to think about an embedment in a larger working context involving further and more detailed transformation and analysis methods. From a *data handling point of view*, it is possible to use design language models for the transition of research concepts to the start of an aircraft project and maybe even further. However, from a *working process point of view*, this means that more people will need to be involved and that the data management efforts will increase. Additionally, the detailed *what* and *when* of the individual working steps may change, as a new analysis as well as new evaluation methods and front-loaded processes can be realized.

[118] While RUDOLPH [131] outlines the relation between computer-aided design and 'artificial intelligence' methods, ERASME [48] can be consulted for an overview about changes and risks related to 'artificial intelligence' approaches.

Currently, there is only limited experience beyond small project teams *programming and using* the CDL models and the analysis results in parallel. The needs for robust ways of working with larger project teams and one central design language-based data model are a research question beyond pure software development and modeling problems.

Especially when it comes to a product project start, where product data and documentation tend to explode, further design language experts[119] and PDM systems must be involved. On one hand, this touches the question of how to involve PDM systems (fig. 6.13). On the other hand, and not by accident, the conference series 'Design Computing and Cognition' (DCC) also involves cognitive, work procedural and even sociological aspects when thinking about design and design computing [56]. Such aspects along with the level of know-how of the involved engineers with respect to design languages play an important role when considering a wider implementation of innovative design methods.

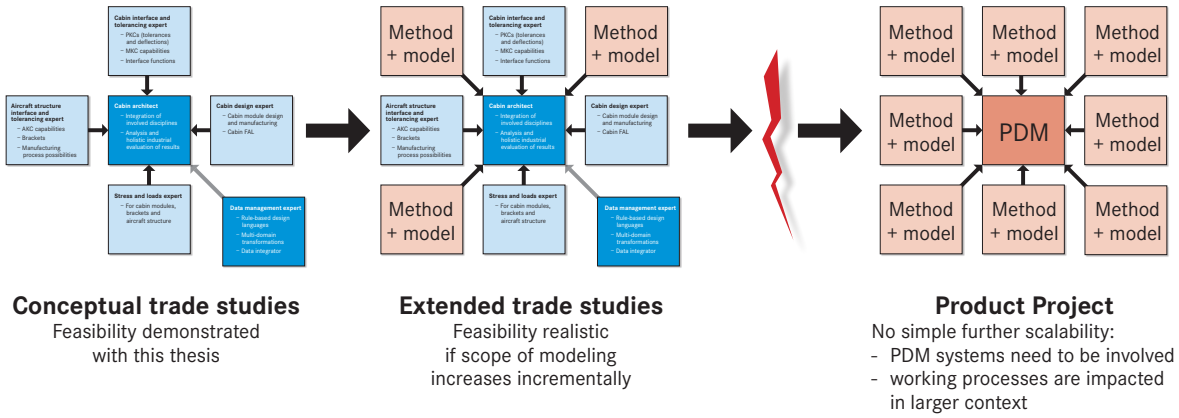

Figure 6.13: *Limited scalability of way of working*

Giving consideration to the *industrial* challenges and limitations mentioned, an extension of the proposed cabin architecture analysis method framework is possible step by step, if the need rises to involve further analysis methods. However, this growth cannot be infinite without implications for the way of working with design languages (fig. 6.13). In particular, this comprises training and working experience with design languages in a wider scope.

[119] See the beginning of this section for a definition of user skill levels.

Chapter 7

Summary

7.1 Results

This thesis copes with physical aspects of aircraft cabin architectures. It sets the focus on conceptual design aspects needed to integrate the cabin into the fuselage (chapter 1). Therefore, tolerancing methods have been outlined (chapter 2) and it shows that tolerancing can play a key role for physical cabin architectures and integration. However, some methodological shortcomings are identified (chapter 3). Firstly there is no clear description of the role of tolerancing for conceptual cabin design, secondly there are no implemented holistic methods available which can support this proceeding in an industrial context.

Within chapter 4, so-called *cabin design languages* (CDL) are introduced to bridge these gaps. Therefore a *scope of modeling* is proposed which is constituted by methods such as analysis of a digital mock-up, tolerancing models used for mechanical interface management, preliminary FAL process planning along with mass and cost estimation methods. Out of this scope of modeling, the multi-domain CDL 'vocabulary' is unfolded. The concepts of *physical components* and of *mechanical interfaces* turned out to be key aspects of the models. Furthermore, the consideration of a physical component as a tolerancing datum system in parallel opens the door to tolerancing methods and to assembly process modeling. The proposed definition of mechanical interface functions is rendered more precisely into classes for toleranced *gaps* (fulfilling a *spatial distance function*), into *kinematic linkages* (fulfilling the *locating function*) and *load interface* (fulfilling the load transmission or *fixation function*). Additionally, several *architecture analysis parameters* (AAPs) are proposed, acting as architecture 'metrics'. Along with the mentioned analysis methods, these AAPs are used to analyze or 'measure' the investigated cabin architectures. To build up the CDL data models, a generic working process based on a tolerance management process is proposed.

Using this preparatory work, an implementation of CDL has been conducted (chapter 5) for the modeling and execution of several important aspects of physical cabin architectures, proving the key role of tolerancing for this task. Firstly, the process chain has been extended by a software plugin to create input files for tolerance analysis software using design languages. Secondly, a plugin containing the CDL class definitions and several analysis plugins have been introduced. The analysis plugins automatically calculate the AAPs, transform CDL models into design language-based tolerance analysis models and compile spreadsheet charts for further analyses. This includes files and models of tolerancing data for exchanging data with the involved fuselage design teams to enable joint 'tolerance design' rather than the unilateral 'top-down definition of tolerances' and their design implications.

A use case demonstrates the application using a segment of a long range aircraft fuselage including some cabin compartment modules. In the sense of design languages, graphical rules have been deployed to compile design model of several trade-off studies comprising the required design aspects, which is followed by automatized model transformations to analysis models.

Chapter 6 shows the technical result of the exemplary use case. It can be demonstrated how data models can be generated fast and effectively for the required analysis task, and how the reproduction of the CDL data models and the various analysis models can be accomplished if input parameters change. Key aspects of the mentioned tolerance management working process could be automatized and it turns out as expected that conceptual tolerancing can be used to support a more holistic definition and analysis of the interface between the cabin and the fuselage. The chosen AAPs prove to be a helpful means to analyze and evaluate technical cabin integration scenarios. This enables a new method of fast trade studies with multi-disciplinary evaluation for physical aspects of cabin architectures including tolerancing methods using data models basing on graphical design languages.

Moreover, it is sketched how further analysis methods such as FEM-based load and stress analysis or functional models of systems could be applied while making use of the abstract UML-based CDL data models. Finally, the repercussions for the way of working for conceptual trade studies in an industrial context have been outlined.

7.2 Outlook

The architecture analysis parameters (AAPs) as introduced within this dissertation represent a pragmatic approach to evaluate cabin architectures. In order to be able to apply these parameters, it is necessary to build up databases for different technical scenarios comparable to those for preliminary aircraft design purposes. Of course, for this data mining effort the proposed approach using design language models can be used as a supporting method. In addition, it is necessary to question the sense of each parameter and to extend the list by further findings. For instance, it may be interesting to investigate some more specific manufacturing-related parameters such as for the description of the work share philosophy between the major component assembly and final assembly line levels.

Heuristics-based methods are often used in engineering to reduce design complexity of huge systems, but tend to become 'unwritten law'. Thus it can be interesting to investigate architecture and design heuristics for cabin integration, which are for instance the convenient cabin compartment module arrangement or the attachment principles. For these purposes, different architecture concepts could be modeled and compared accordingly. On top of that, cabin customization aspects have not yet been considered in detail. Even though it is possible to model different customer layouts, there could be a certain need to have a deeper look into these aspects.

A part of this work's focus is to describe the term 'mechanical interface' in the context of physical aircraft cabin integration. The proposed concept of functional separation between *spatial distance function*, *locating function* and *fixation function* does not claim to be generally valid. Further research could be directed on a more general definition of physical and mechanical interfaces, which is a 'meeting place' for many analysis methods for architectures and conceptual design. Due to the modular concept of graph-based design languages, it is possible to exchange or extend the used 'vocabulary' with more detailed concepts. For instance, the functional geometry features used to model the mechanical interfaces between physical components could be extended to parameter-based geometry descriptions such as three-dimensional splines or shapes.

With the current research status, it would still be difficult to automatize the definition of complex tolerancing datum systems for general purposes. The decisions rules for datum system generation and optimizations are complex and even within a single field of application like aircraft cabin tolerancing no generally valid rules could be specified until now. The automatization of entire iteration loops for complex tolerance optimization processes such that there is a benefit in time and cost for engineering at short notice is also still difficult. However, these aspects have not been in the focus of this thesis and the research gap could be closed with future work.

But despite the challenge to express 'generally valid' design rules, it is possible to direct further research in the abstraction of cabin architectures and design, making use of the achievements of the presented *cabin design language* (CDL) approach.

At this point, extending or complementing the CDL models towards aircraft systems can come into focus. This can lead to considering approaches using design languages for a wider scope beyond specific physical architecture tasks, such as aircraft systems analysis or even detailed design aspects. A further question in this respect is the potential link or exchange of design language models with PDM or PLM systems. Additionally, it showed that using design languages as design support method requires specific skills of the involved engineers, which may differ to those needed for conventional engineering means. Three different levels of user skills have been named, which rises the need for specific trainings and for efforts to integrate design processes using design languages into existing ways of working. However, these investment may pay of well in the long term or to say it in other words: *'there is no such thing as a free lunch'*[120] when engineering complex systems.

[120] See http://en.wikipedia.org/wiki/There_ain't_no_such_thing_as_a_free_lunch, accessed Feb 2012.

Appendix A

Symbols for Tolerance Specification

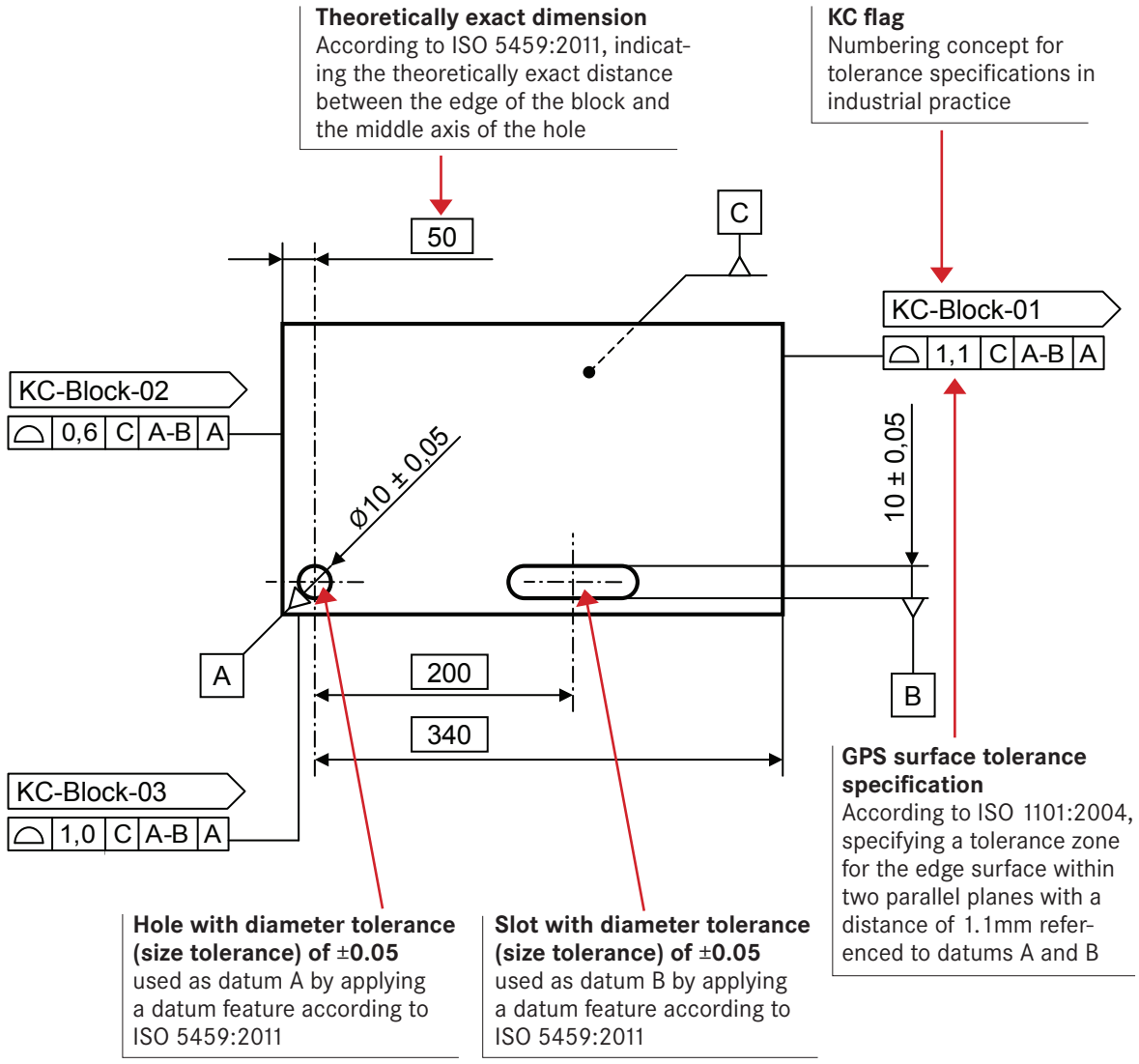

Figure A.1: *Explanations and references for the tolerance specification symbols shown in fig. 2.1, page 20*

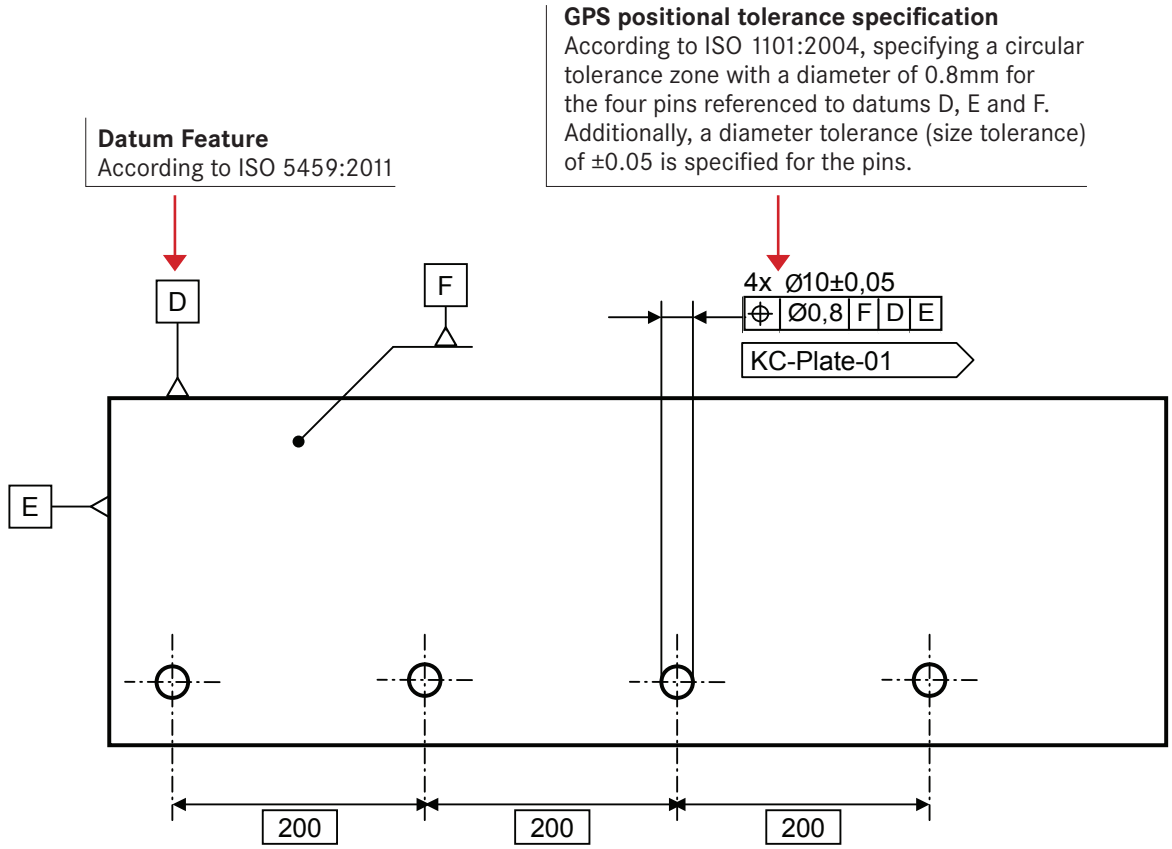

Figure A.2: *Explanations and references for the tolerance specification symbols shown in fig. 2.1, page 20*

Appendix B

CDL Class Diagram

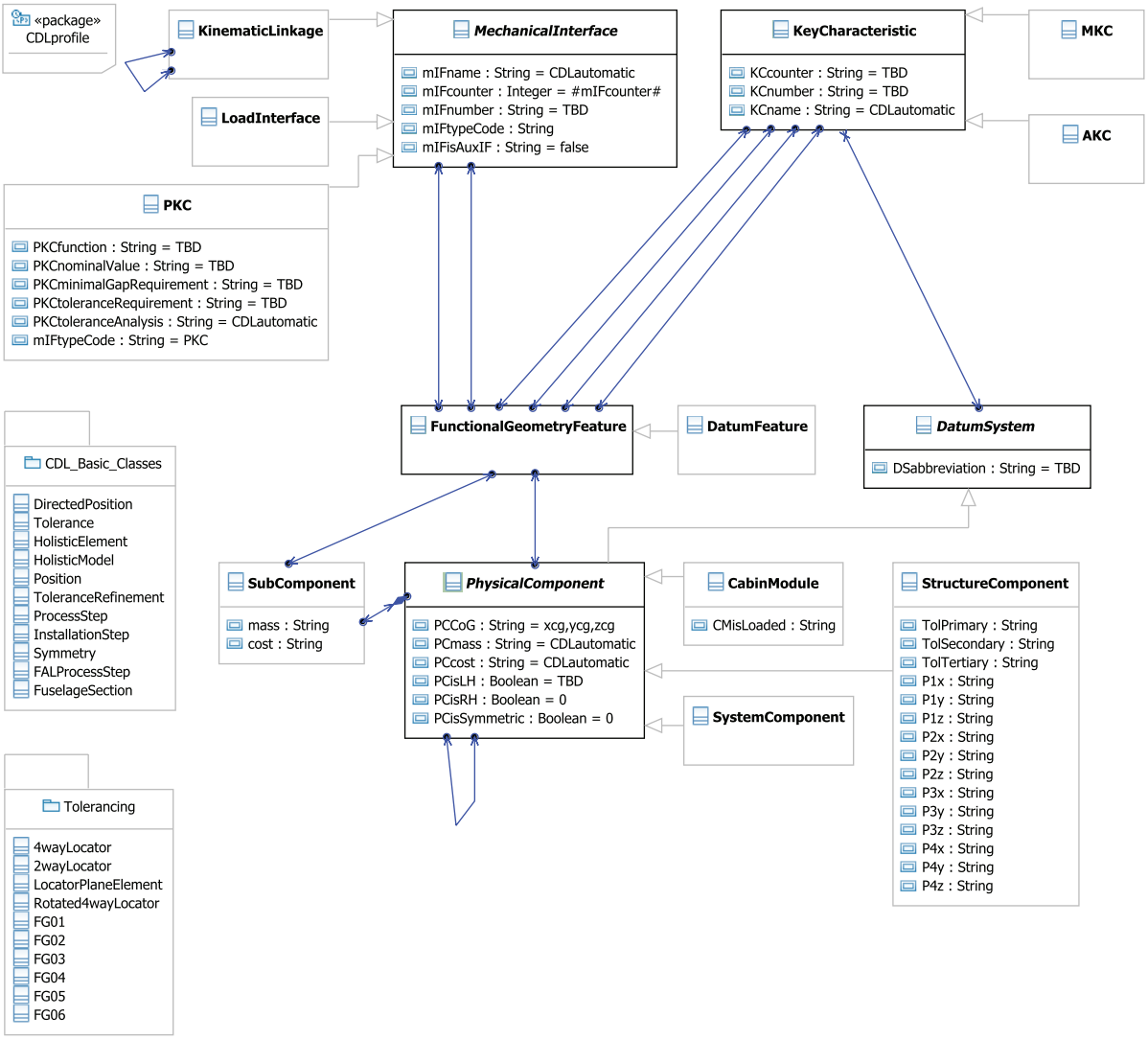

Figure B.1: *The full CDL class diagram in UML from the plugin de.bl.cdl*

Class name	PhysicalComponent
Description	Physical entity or system in terms of systems engineering. Contains related architecture parameters (e.g., positional data, size parameters), manufacturing data (e.g., production capabilities, cost). Also contains parameters for the description of the installation or assembly process.
Model interfaces	SubComponent, FunctionalGeometryFeature
Inherits from	HolisticElement, DatumSystem, ProcessStep, Position
Inheritors	CabinModule, StructureComponent, SystemComponent
Operations	-

Figure B.2: *CDL class 'PhysicalComponent'*

Class name	DatumSystem
Description	Reference system of a PhysicalComponent used in the sense of ISO 5459:2011 for tolerancing purposes. Consists of a primary, a secondary and a tertiary datum plane, which are not always modeled explicitly. If DatumFeature objects are modeled explicitly for a PhysicalComponent, the model needs to be checked for pragmatic correctness.
Model interfaces	DatumFeature [0..*]
Inherits from	-
Inheritors	PhysicalComponent
Operations	-

Figure B.3: *CDL class 'DatumSystem'*

Class name	ProcessStep
Description	Installation or assembly process in order to install a PhysicalComponent. Contains the preparation and installation time and cost as attributes and can consist of multiple individual InstallationStep objects.
Model interfaces	InstallationStep [0..*]
Inherits from	-
Inheritors	PhysicalComponent
Operations	Calculation of slot for installation time/cost (PSInstallationTime/PSInstallationCost) as sum of installation time/cost of the corresponding InstallationSteps

Figure B.4: *CDL class 'ProcessStep'*

Class name	CabinModule
Description	Special PhysicalComponent fulfilling one or more cabin functions.
Model interfaces	-
Inherits from	PhysicalComponent
Inheritors	Scenario-specific classes
Operations	Calculation of slots for mass/cost (PCmass/PCcost) as sum of SubComponent masses/costs. Definition of functional directions of corresponding FunctionalGeometryFeatures according to the functional directions of the linked MIFs. If necessary, the coordinates and directions are transformed into the coordinate system of the CabinModule. Check if the explicit datum definition using DatumFeatures is valid according to ISO 5459:2011 (the functional directions are compared and it is checked, if the constitute independent vectors to span up a 3-2-1 system). If not, a error message is created in the error log file. Check, if the linkage system is isostatic (the functional directions are compared and it is checked if the constitute independent vectors to span up a 3-2-1 system). If not, an error message is created in the error log file.

Figure B.5: *CDL class 'CabinModule'*

Class name	SystemComponent
Description	Special PhysicalComponent fulfilling one or more aircraft system functions.
Model interfaces	-
Inherits from	PhysicalComponent
Inheritors	Scenario-specific classes
Operations	-

Figure B.6: *CDL class 'SystemComponent'*

Class name	StructureComponent
Description	Special PhysicalComponent fulfilling one or more aircraft structure functions. Acts as integration datum system for structure tolerances. Links FunctionalGeometryFeatures of the aircraft structure.
Model interfaces	-
Inherits from	PhysicalComponent
Inheritors	Scenario-specific classes
Operations	Calculation of slots for mass/cost (PCmass/PCcost) as sum of SubComponent masses/costs

Figure B.7: *CDL class 'StructureComponent'*

Class name	SubComponent
Description	3D geometry data for DMU visualization and mass data estimation corresponding to a PhysicalComponent including all required parameters.
Model interfaces	Special links to other SubComponent of FunctionalGeometryFeature objects
Inherits from	HolisticElement, Position
Inheritors	Scenario-specific classes
Operations	-

Figure B.8: *CDL class 'SubComponent'*

Class name	MechanicalInterface
Description	Interconnection between two FunctionalGeometryFeature objects belonging to two different PhysicalComponents. Serves a dedicated functional purpose described explicitly by the inheritor classes.
Model interfaces	FunctionalGeometryFeatures
Inherits from	InstallationStep, HolisticElement, DirectedPosition
Inheritors	PKC, KinematicLinkage, LoadLinkage
Operations	-

Figure B.9: *CDL class 'MechanicalInterface'*

Class name	InstallationStep
Description	Single working step during the installation process of a PhysicalComponent with the corresponding working time and cost.
Model interfaces	ProcessStep
Inherits from	-
Inheritors	MechanicalInterface
Operations	-

Figure B.10: *CDL class 'InstallationStep'*

Class name	PKC (Abbr. for 'Performance Key Characteristic')
Description	A static tolerance requirement according to ISO 286:2010 nomenclature. Fulfills a 'spatial distance function' (see ch. 4.2.2). Additionally can be a dynamic in-flight deflection limitation requirement between two FunctionalGeometryFeature objects. Serves a dedicated function indicating the functional quality (e.g. optical requirement for a gap, minimum-distance requirement, installability check).
Model interfaces	(see MechanicalInterface)
Inherits from	-
Inheritors	-
Operations	Definition of naming in slot MIFname according to nomenclature. Definition of coordinates as center point of the two corresponding FunctionalGeometry-Features.

Figure B.11: *CDL class 'PKC'*

Class name	KinematicLinkage
Description	Functional kinematic relation between two FunctionalGeometryFeature objects applying restrictions to the kinematic degrees of freedom between these objects. Fulfills a 'locating function' (see ch. 4.2.2). Establishes a linkage system for the corresponding PhysicalComponents.
Model interfaces	(see MechanicalInterface)
Inherits from	MechanicalInterface
Inheritors	2wayLocator, 4wayLocator, Rotated4wayLocator, LocatorPlaneElement
Operations	Definitions of naming in slot MIFname according to nomenclature. Definition of coordinates according to those of the FunctionalGeometryFeatures of the corresponding CabinModule (coordinates in the local CabinModule coordinate system are transformed into global coordinates).

Figure B.12: *CDL class 'KinematicLinkage'*

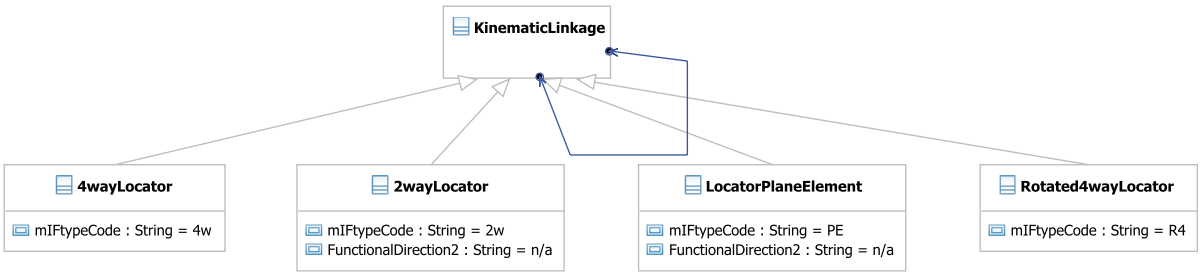

Figure B.13: *Special 'KinematicLinkage' classes*

Class name	LoadInterface
Description	Functional mechanic relation between two FunctionalGeometryFeature objects enabling static or dynamic force or moment transfer between these objects. Fulfills a 'fixation function' (see ch. 4.2.2). Often coupled with KinematicLinkage.
Model interfaces	(see MechanicalInterface)
Inherits from	MechanicalInterface
Inheritors	-
Operations	same as for KinematicLinkages

Figure B.14: *CDL class 'LoadInterface'*

Class name	FunctionalGeometryFeature
Description	Geometry feature of a PhysicalComponent with a functional purpose needed to realize a mechanical interface to another FunctionalGeometryFeature of another PhysicalComponent. Can be toleranced by a KeyCharacteristic or can be declared as DatumFeature of the corresponding PhysicalComponent respectively DatumSystem. Can be linked to a SubComponent for visualization and parameter calculation purposes.
Model interfaces	PhysicalComponent, MechanicalInterface, KeyCharacteristic, SubComponent
Inherits from	DirectedPosition
Inheritors	Library of standard features (e.g., point, plane, line, circle), if required
Operations	Routines to define coordinates and functional direction depending on the associated PhysicalComponents and MechanicalInterfaces

Figure B.15: *CDL class 'FunctionalGeometryFeature'*

Class name	DatumFeature
Description	Specialization for a FunctionalGeometryFeature forming part of the Datum-System of the corresponding PhysicalComponent. Used in the sense of ISO 5459:2011.
Model interfaces	-
Inherits from	-
Inheritors	-
Operations	-

Figure B.16: *CDL class 'DatumFeature'*

Class name	KeyCharacteristic
Description	Used in the sense of EN 9100:2010 for special positional attributes of a FunctionalGeometryFeature, which have influence on architecture evaluation parameters.
Model interfaces	DatumSystem
Inherits from	Tolerance
Inheritors	AKC and MKC as special KeyCharacteristics for CabinModules and aircraft StructureComponents
Operations	Special routines for MKCs and AKCs to set naming in slot KCName according to nomenclature

Figure B.17: *CDL class 'KeyCharacteristic'*

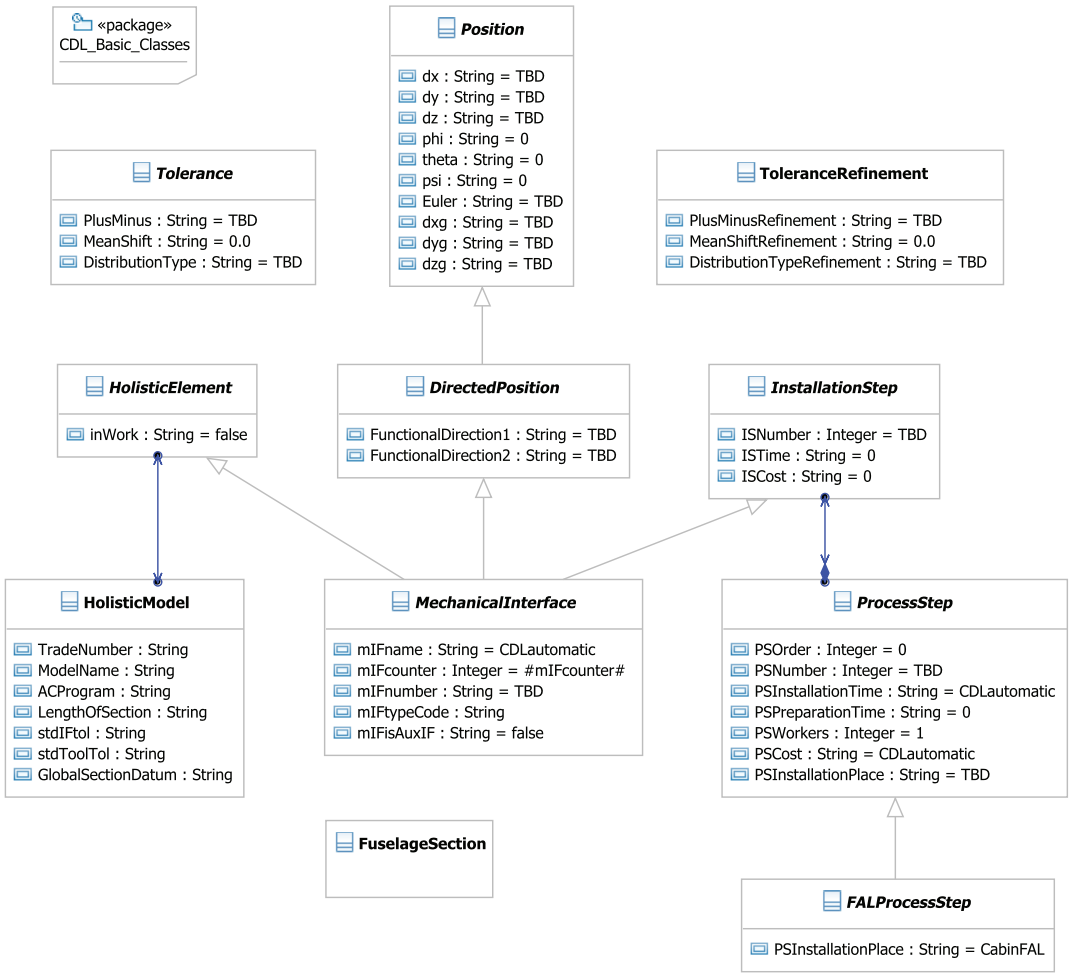

Figure B.18: *CDL class diagram 'Basics'*

Class name	HolisticModel
Description	Top node or placeholder for the whole model. Contains global model parameters, which have to be available to any model element classified as HolisticElement. Acts as global coordinate system for the model. Only one instance per scenario (instance is referred to as HMinst).
Model interfaces	To any instance of classifier HolisticElement
Inherits from	-
Inheritors	Scenario-Specific class with individualized and extended global parameters
Operations	-

Figure B.19: *CDL class 'HolisticModel'*

Class name	HolisticElement
Description	Abstract classifier for any multi-disciplinary CDL model element, which needs a link to the HMinst.
Model interfaces	To HMinst
Inherits from	-
Inheritors	PhysicalComponent, SubComponent, MechanicalInterface
Operations	-

Figure B.20: *CDL class 'HolisticElement'*

Class name	Position
Description	Set of Cartesian coordinates and Euler angles to describe the position and the orientation of a geometry object relative to a coordinate system.
Model interfaces	-
Inherits from	-
Inheritors	PhysicalComponent, SubComponent, DirectedPosition
Operations	Transform local coordinates into global coordinates upon need

Figure B.21: *CDL class 'Position'*

Class name	DirectedPosition
Description	Special position definition describing the functional direction(s) for directed functional model elements.
Model interfaces	-
Inherits from	-
Inheritors	MechanicalInterface, FunctionalGeometryFeature
Operations	Transform local coordinates and functional directions into global ones upon need

Figure B.22: *CDL class 'DirectedPosition'*

Class name	Tolerance
Description	Used in the sense of ISO 286:2010. Contains tolerance value, mean shift value and statistical distribution information.
Model interfaces	-
Inherits from	-
Inheritors	KeyCharacteristic
Operations	Check if the slot PlusMinus contains a valid value (signed number or number with ±-sign). If not, an error message is created in the error log file.

Figure B.23: *CDL class 'Tolerance'*

Class name	ToleranceRefinement
Description	Similar to Tolerance, except for the fact that it represents a group tolerance according to ISO 1101:2004
Model interfaces	-
Inherits from	-
Inheritors	KeyCharacteristic
Operations	Check if the slot PlusMinus contains a valid value (signed number or number with ±-sign). If not, an error message is created in the error log file.

Figure B.24: *CDL class 'ToleranceRefinement'*

Appendix C

CDL_LR Class Diagram

Figure C.1: *The full CDL_LR class diagram in UML*

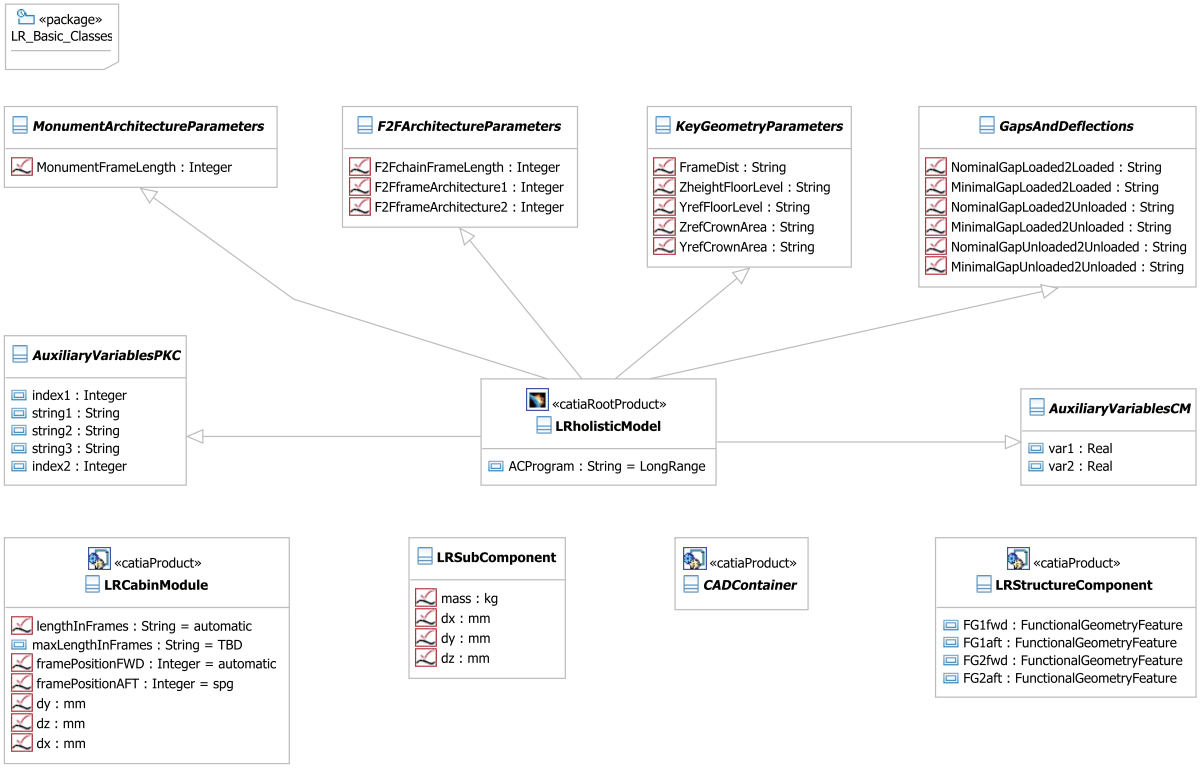

Figure C.2: *The 'basic classes' of the CDL_LR class diagram in UML. 'LRCabinModule', 'LRSubComponent' and 'LRStructureComponent' inherit from 'CabinModule', 'StructureComponent' and 'CabinModule' respectively.*

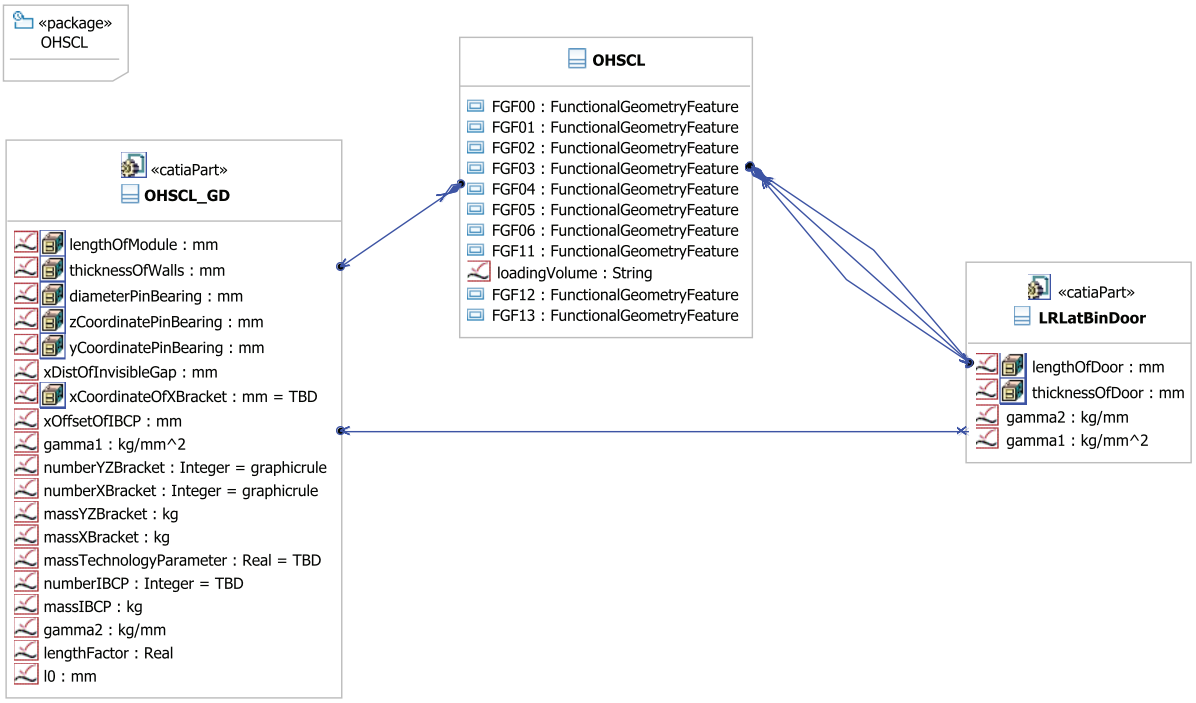

Figure C.3: *The 'lateral OHSC classes' of the CDL_LR class diagram in UML. 'OHSCL' inherits from 'LRCabinModule', the others inherit from 'LRSubComponent'.*

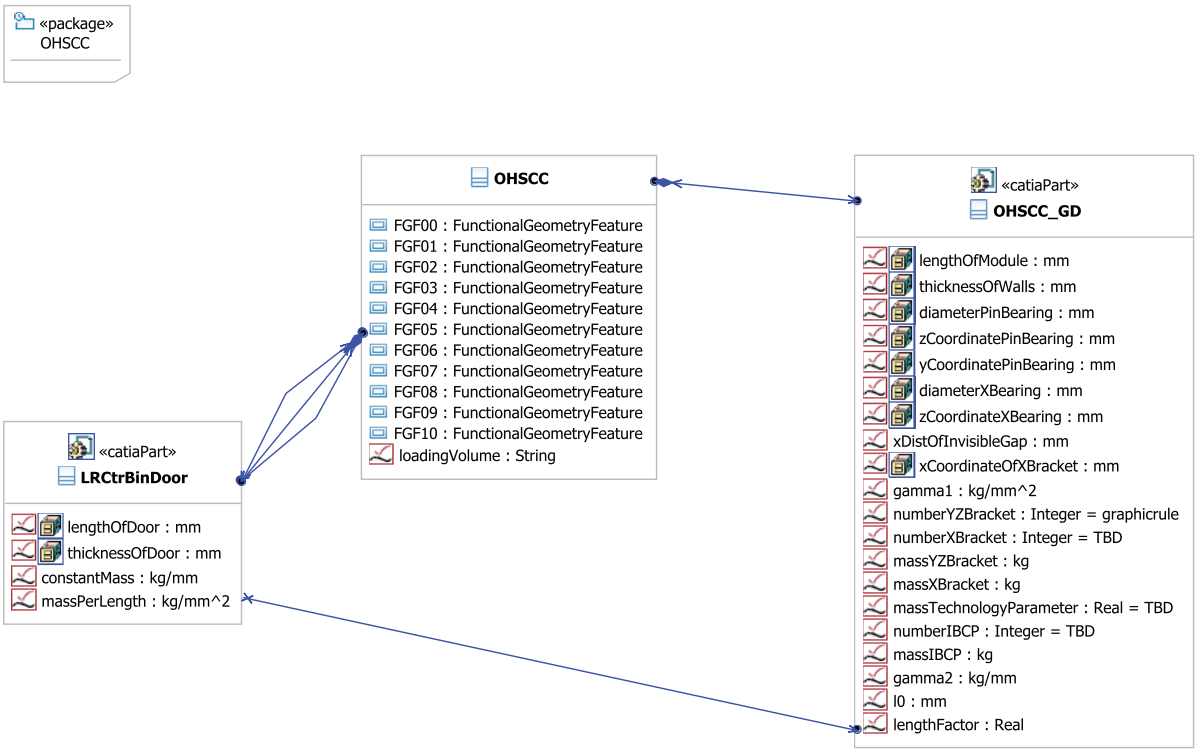

Figure C.4: *The 'center OHSC classes' of the CDL_LR class diagram in UML. 'OHSCC' inherits from 'LRCabinModule', the others inherit from 'LRSubComponent'.*

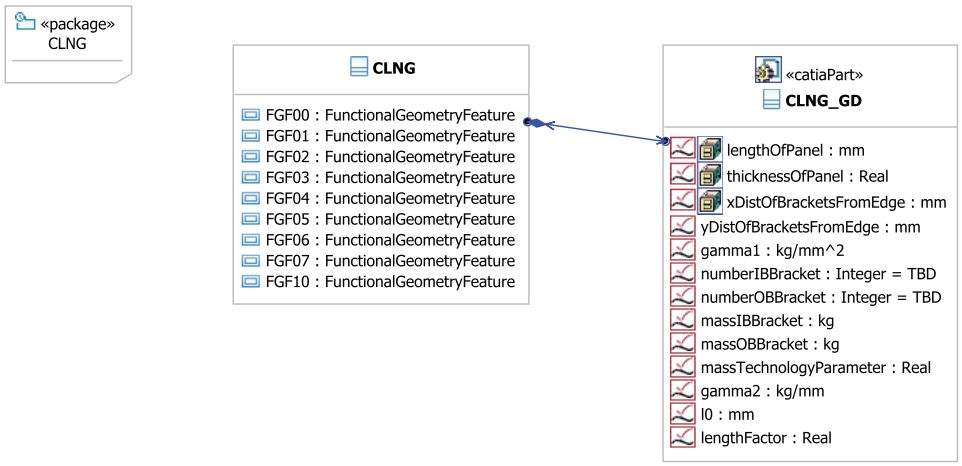

Figure C.5: *The 'ceiling panel classes' of the CDL_LR class diagram in UML. 'CLNG' inherits from 'LRCabinModule', 'CLNG_GD' inherits from 'LRSubComponent'.*

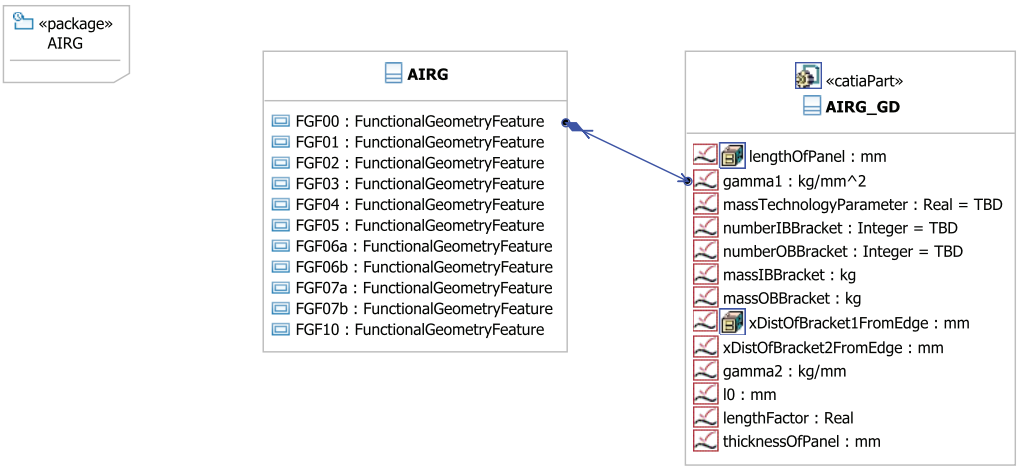

Figure C.6: *The 'air grid classes' of the CDL_LR class diagram in UML. 'AIRG' inherits from 'LRCabinModule', 'AIRG_GD' inherits from 'LRSubComponent'.*

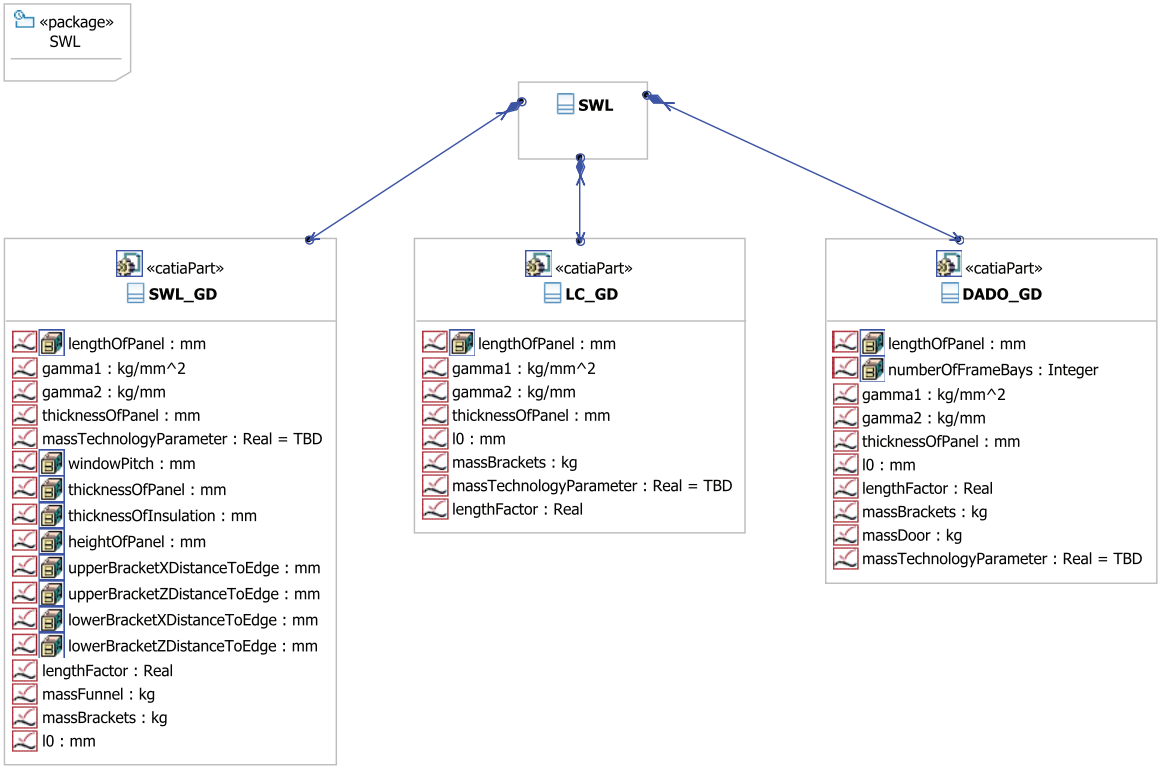

Figure C.7: *The 'sidewall lining classes' of the CDL_LR class diagram in UML. 'SWL' inherits from 'LRCabinModule', the others inherit from 'LRSubComponent'.*

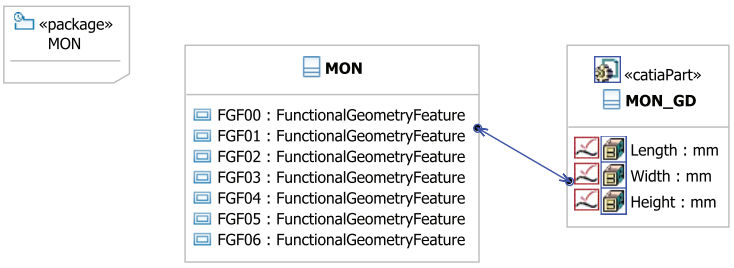

Figure C.8: *The 'monument classes' of the CDL_LR class diagram in UML. 'MON' inherits from 'LRCabinModule', 'MON_GD' inherits from 'LRSubComponent'.*

Figure C.9: *The 'aircraft structure classes' of the CDL_LR class diagram in UML. The top five classes inherit from 'LRStructureComponent', the others inherit from 'LRSubComponent'.*

Bibliography

[1] ABULAWI, J. Ansätze zur Beherrschung der Eigenkomplexität von parametrisch-assoziativen CAD-Modellen. In *Proceedings of the Conference Go-3D, 2011, Rostock* (Stuttgart, 2011), Fraunhofer Verlag, pp. 137–150.

[2] ABULAWI, J., SEECKT, K., POMMERS, M., AND SCHOLZ, D. Automatic generation of 3D-CAD models to bridge the gap between aircraft preliminary sizing and geometric design. In *DGLR 60. Deutscher Luft- und Raumfahrtkongress (DLRK), 2011, Bremen, Tagungsband - Manuskripte (CD), No. 241195* (2011), pp. 405–415.

[3] AIRLINES FOR AMERICA (A4A). iSpec 2200 extract: ATA standard numbering system. product code 11800p, revision 2011.1. Internet, https://publications.airlines.org/, accessed Jan 2012, 2011.

[4] ALBANO, L. D., AND SUH, N. P. Axiomatic approach to structural design. *Research in Engineering Design 4* (1992), 171–183.

[5] ALBER, R., RUDOLPH, S., AND KRÖPLIN, B. On formal languages in design generation and evolution. In *Proceedings CD of the Fifth World Congress on Computational Mechanics (WCCM), 2002, Vienna* (2002).

[6] AMETA, G., DAVIDSON, J. K., AND SHAH, J. J. Tolerance-maps applied to a point-line cluster of features. *Journal of Mechanical Design 129* (2007), 782–792.

[7] ANDERSSON, H. *Aircraft systems modeling – model based systems engineering in avionics design and aircraft simulation*. Dissertation, Linköping University, 2010.

[8] ANTONSSON, E. K., AND CAGAN, J. *Formal engineering design synthesis*. Cambridge University Press, 2005.

[9] ARMSTRONG, C. G., MONAGHAN, D. J., PRICE, M. A., OU, H., AND LAMONT, J. Integrating CAE concepts with CAD geometry. In *Engineering Computational Technology – Proceedings of the 3rd International Conference on Engineering Computational Technology, 2002, Prague* (Stirling, 2002), Saxe-Coburg Publications, pp. 75–104.

[10] ARNOLD, P., AND RUDOLPH, S. Bridging the gap between product design and product manufacturing by means of graph-based design languages. In *Proceedings of the 9th International Symposium on Tools and Methods of Competitive Engineering TMCE, 2012, Karlsruhe* (2012).

[11] BAILEY, J., AND CLARK, N. Parts didn't click together for Boeing jet. Published in the Internet, January 17, 2008, http://www.nytimes.com/2008/01/17/business/17plane.html?_r=1, accessed Jan 2012, 2008.

[12] BALLU, A., FALGARONE, H., CHEVASSUS, N., AND MATHIEU, L. A new design method based on functions and tolerance specifications for product modelling. *CIRP Annals – Manufacturing Technology 55*, 1 (2006), 139–142.

[13] BALLU, A., PLANTEC, J. Y., AND MATHIEU, L. Geometrical reliability of overconstrained mechanisms with gaps. *CIRP Annals – Manufacturing Technology 57*, 1 (2008), 159–162.

[14] BERGHOLZ, M. *Objektorientierte Fabrikplanung*. Dissertation, RWTH Aachen, 2005.

[15] BÖHNKE, D., LITZ, M., NAGEL, B., AND RUDOLPH, S. Evaluation of modeling languages for preliminary airplane design in multidisciplinary design environments. In *DGLR Deutscher Luft- und Raumfahrtkongress (DLRK), 2010, Hamburg* (2010).

[16] BÖHNKE, D., REICHWEIN, A., AND RUDOLPH, S. Design language for airplane geometries using the unified modeling language. In *Proceedings of the ASME International Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE), 2009, San Diego, California, DETC2009-87368* (2009), American Society of Mechanical Engineers.

[17] BOURDET, P., AND CLÉMENT, A. Controlling a complex surface with a 3 axis measuring machine. *Annals of the CIRP 25*, 1 (1976), 359–361.

[18] BRONSTEIN, I. N., SEMENDJAJEW, K. A., MUSIOL, G., AND MÜHLIG, H. *Taschenbuch der Mathematik, 7. Auflage*. Verlag Harri Deutsch, Frankfurt am Main, 2001.

[19] CHASE, K. W. Basic tools for tolerance analysis of mechanical assemblies. In *Manufacturing engineering handbook* (2004), McGraw-Hill.

[20] CHASE, K. W., GAO, J., AND MAGLEBY, S. P. General 2-D tolerance analysis of mechanical assemblies with small kinematic adjustments. *Journal of Design and Manufacturing 5*, 4 (1995), 263–274.

[21] CHASE, K. W., GAO, J., MAGLEBY, S. P., AND SORENSON, C. D. Including geometric feature variations in tolerance analysis of mechanical assemblies. *IIE Transactions 28* (1996), 795–807.

[22] CHASE, K. W., AND GREENWOOD, W. H. Design issues in mechanical tolerance analysis. *ASME Manufacturing Review 1*, 1 (1988), 50–59.

[23] CHASE, K. W., MAGLEBY, S. P., AND GLANCY, C. G. A comprehensive system for computer-aided tolerance analysis of 2d and 3d mechanical assemblies. In *Proceedings of the 5th CIRP Seminar on Computer-Aided Tolerancing, 1997, Toronto* (1997).

[24] CHAU, H. H., CHEN, X., MCKAY, A., AND DE PENNINGTON, A. Evaluation of a 3D shape grammar implementation. In *Design Computing and Cognition '04 – Proceedings of the 1st International Conference on Design Computing and Cognition (DCC'04), 2004, Cambridge, Massachusetts* (Dordrecht, 2004), Kluwer Academic Publishers, pp. 357–376.

[25] CHOUDRI, A. Design for Six Sigma for aerospace applications. In *AIAA Space Conference and Exposition, 2004, San Diego, California AIAA 2004-6127* (2004), American Institute of Aeronautics and Astronautics, pp. 2402–2408.

[26] CLÉMENT, A., AND BOURDET, P. A study of optimal-criteria identification based on the small-displacement screw model. *CIRP Annals – Manufacturing Technology 37*, 1 (1988), 503–506.

[27] CLOZEL, P. 3D tolerances analysis, from preliminary study. In *Proceedings of the 7th CIRP Seminar on Computer-Aided Tolerancing, 2001, Cachan* (2003), pp. 93–104.

[28] CLOZEL, P., AND RANCE, P. A. MECAmaster: a tool for assembly simulation from early design, industrial approach. In *Geometric Tolerancing of Products* (2010), Wiley ISTE, pp. 241–272.

[29] CONCHERI, G., CRISTOFOLINI, I., MENEGHELLO, R., AND WOLF, G. Geometric Dimensioning and Tolerancing (GD&T) versus Geometrical Product Specification (GPS). In *Proceedings of the XII ADM International Conference on Design Tools and Methods in Industrial Engineering, 2001, Rimini* (2001).

[30] CRAWLEY, E., DE WECK, O., EPPINGER, S., MAGEE, C., MOSES, J., SEERING, W., SCHINDALL, J., WALLACE, D., AND WHITNEY, D. The influence of architecture in engineering systems. *Engineering Systems Monograph, Massachusetts Institute of Technology (MIT)* (2004).

[31] CUMMINGS, T. K. Lessons learned from 737 & 787 jet liner programs. In *AIAA Space Conference and Exposition, 2007, Long Beach, California, AIAA 2007-6125* (2007), American Institute of Aeronautics and Astronautics.

[32] CURRAN, R., KUNDU, A., RAGHUNATHAN, S., EAKIN, D., AND MCFADDEN, R. Influence of manufacturing tolerance on aircraft direct operating cost (DOC). *Journal of Materials Processing Technology 138* (2003), 208–213.

[33] CZARNECKI, K., AND HELSEN, S. Classification of model transformation approaches. In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, 2003, Anaheim, California* (2003).

[34] DANTAN, J. Y., ANWER, N., AND MATHIEU, L. Integrated tolerancing process for conceptual design. *CIRP Annals – Manufacturing Technology 52*, 1 (2003), 135–138.

[35] DANTAN, J. Y., AND BALLU, A. Assembly specification by gauge with internal mobilities (GIM) – Specification semantics deduced from tolerance synthesis. *Journal of Manufacturing Systems 21*, 3 (2002), 218–235.

[36] DANTAN, J. Y., BALLU, A., AND MARTIN, P. Mathematical formulation of tolerance synthesis taking into account the assembly process. In *Proceedings of the 5th IEEE International Symposium on Assembly and Task Planning, 2003, Besançon* (2003), pp. 241–246.

[37] DANTAN, J. Y., BALLU, A., AND MATHIEU, L. Geometrical product specifications – model for product life cycle. *Computer-Aided Design 40* (2008), 493–501.

[38] DANTAN, J. Y., MATHIEU, L., BALLU, A., AND MARTIN, P. Tolerance synthesis: quantifier notion and virtual boundary. *Computer-Aided Design 37* (2005), 231–240.

[39] DAVIDSON, J. K., AND SHAH, J. J. Mathematical model to formalize tolerance specifications and enable full 3D tolerance analysis. In *Proceedings of NSF Design, Service and Manufacturing Grantees and Research Conference (SMU), 2004, Dallas, Texas* (2004).

[40] DEPARTMENT OF DEFENCE – SYSTEMS MANAGEMENT COLLEGE. *Systems engineering fundamentals*. Defense Aquisition University Press, Fort Belvoir, Virginia, 2001.

[41] DIMENSIONAL CONTROL SYSTEMS. Dimensional engineering news, issue 70, February 2009. Newsletter, subscription using http://www.3dcs.com/pages/mailinglist.php, accessed Jan 2012.

[42] DIMENSIONAL CONTROL SYSTEMS. 3DCS software help manual '3DCS Analyst, CAA V5 Based', version 6.18.0.0.0. Released with 3DCS software distribution, Dimensional Control Systems, Inc., Troy, Michigan, 2009.

[43] EARLY, J., PRICE, M., CURRAN, R., RAGHUNATHAN, S., BENARD, E., AND CASTAGNE, S. Constraint management for engineering systems. In *5th AIAA Aviation Technology, Integration and Operations (ATIO) Forum, 2005, Washington, D.C. AIAA 2005-7374* (2005), American Institute of Aeronautics and Astronautics.

[44] EARLY, J. M., PRICE, M., MAWHINNEY, P., CURRAN, R., AND RAGHUNATHAN, S. Framework tools for modelling and simulation in an integrated design environment. In *4th AIAA Aviation Technology, Integration and Operations (ATIO) Forum, 2004, Chicago, Illinois, AIAA 2004-6457* (2004), American Institute of Aeronautics and Astronautics.

[45] EHRLENSPIEL, K. *Integrierte Produktentwicklung – Denkabläufe, Methodeneinsatz, Zusammenarbeit*, 2nd ed. Carl Hanser Verlag, München – Wien, 2003.

[46] EHRLENSPIEL, K., KIEWERT, A., AND LINDEMANN, U. *Kostengünstig Entwickeln und Konstruieren, Kostenmanagement bei der integrierten Produktentwicklung*, 5th ed. Springer-Verlag, Berlin – Göttingen – Heidelberg, 1999.

[47] EIGNER, M., AND STELZER, R. *Produktdatenmanagement-Systeme: Ein Leitfaden für Product Development und Life Cycle Management*, 2nd ed. Springer-Verlag, Berlin – Heidelberg, 2009.

[48] ERASSME, R. *Der Mensch und die „Künstliche Intelligenz" – eine Profilierung und kritische Bewertung der unterschiedlichen Grundauffassungen vom Standpunkt des gemäßigten Realismus*. Dissertation, RWTH Aachen, 2002.

[49] FALGARONE, H., AND CHEVASSUS, N. An innovative design method and tool for structural and functional analysis. In *Proceedings of the 14th International CIRP Design Seminar, 2004, Kairo* (2004).

[50] FALGARONE, H., AND CHEVASSUS, N. Structural and functional analysis for assemblies. In *Advances in Design* (London, 2006), Springer-Verlag, pp. 87–96.

[51] FARMER, L. E., AND HARRIS, A. G. Change of datum of the dimensions on engineering design drawings. *International Journal of Machine Tool Design & Research 24*, 4 (1984), 267–275.

[52] GAO, J., CHASE, K. W., AND MAGLEBY, S. P. Comparison of assembly tolerance analysis by the direct linearization and modified Monte Carlo simulation methods. In *Proceedings of the ASME Design Engineering Technical Conferences (DETC), 1995, Boston, Massachusetts, DE-Vol. 82* (1995), American Society of Mechanical Engineers, pp. 353–360.

[53] GEDENRYD, H. *How Designers Work*. Dissertation, Lund University, 1998.

[54] GERMER, C. Statistische Toleranzanalyse von beweglichen Karosserie-Anbauteilen. *Konstruktion*, 7/8 (2010), 44–47.

[55] GERMER, C., AND FRANKE, H. J. Interdisziplinäres Toleranzmanagement. In *Proceedings of the 14th Symposium 'Design for X', 2003, Neukirchen* (2003).

[56] GERO, J. S., Ed. *Design Computing and Cognition '10 – Proceedings of the 4th International Conference on Design Computing and Cognition (DCC'10), 2010, Stuttgart* (2011), Springer Science + Business Media.

[57] GÖPFERT, J. *Modulare Produktentwicklung: zur gemeinsamen Gestaltung von Technik und Organisation*. BoD – Books on Demand, Norderstedt, 2009.

[58] GÖPFERT, J., AND STEINBRECHER, M. Modulare Produktentwicklung leistet mehr: warum Produktarchitektur und Projektorganisation gemeinsam gestaltet werden müssen. *Harvard Business Manager*, 3 (2000), 20–48.

[59] GROSS, J., REICHWEIN, A., RUDOLPH, S., BOCK, D., AND LAUFER, R. An executable unified product model based on UML to support satellite design. In *AIAA Space Conference and Exposition, 2009, Pasadena, California, AIAA 2009-6642* (2009), American Institute of Aeronautics and Astronautics.

[60] GRUHN, V., PIEPER, D., AND RÖTTGERS, C. *MDA: effektives Software-Engineering mit UML2 und Eclipse*. Springer Verlag, Berlin – Heidelberg, 2006.

[61] HALFMANN, N., AND KRAUSE, D. Entwicklung montagegerechter Flugzeugkabinen. In *Proceedings of the 20th Symposium 'Design for X', 2009, Neukirchen* (2009), pp. 39–48.

[62] HAQ, M., AND RUDOLPH, S. „EWS-CAR“: eine Entwurfssprache für den Fahrzeugentwurf. *VDI-Berichte*, 1846 (2004), 213–237.

[63] HAQ, M., AND RUDOLPH, S. A design language for generic space-frame structure design. *International Journal of Computer Applications in Technology 30*, 1 (2007), 77–87.

[64] HARMSEN, N., KORMEIER, T., AND RUDOLPH, S. Structural dynamic conceptual design of satellites by design language. In *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Honolulu, Hawaii, 2007, AIAA 2007-2285* (2007), American Institute of Aeronautics and Astronautics.

[65] HAYES, C. C., AND SUN, H. Modeling manufacturing tolerances for process planning of aerospace parts. In *The AIAA 9th Computing in Aerospace Conference, 1993, AIAA-93-4685-CP* (1993), American Institute of Aeronautics and Astronautics, pp. 1224–1230.

[66] HEINZE, W., ÖSTERHELD, C. M., AND HORST, P. Multidisziplinäres Flugzeugentwurfsverfahren PrADO – Programmentwurf und Anwendung im Rahmen von Flugzeug-Konzeptstudien. In *DGLR Deutscher Luft- und Raumfahrtkongress (DLRK), 2001, Hamburg, DGLR-JT2001-194* (2001).

[67] HEISSERMAN, J. A design representation to support automated design generation. In *Proceedings of Artificial Intelligence in Design Conference AID '00, 2000, Worcester, Massachusetts* (2000), Kluwer Academic Publishers, pp. 545–566.

[68] HEISSERMAN, J., AND WOODBURY, R. Generating languages of solid models. In *Proceedings of the Second ACM Symposium on Solid Modeling and Applications, 1993, Montreal* (1993), pp. 103–113.

[69] HENZOLD, G. *Form und Lage*. Beuth-Kommentare. Beuth-Verlag, Berlin, 1999.

[70] HILLSTRÖM, F. Applying axiomatic design to interface analysis in modular product development. *ASME Advances in Design Automation, DE-Vol. 69-2* (1994), 363–371.

[71] HOCHMUTH, R. *Methoden und Werkzeuge als Teil eines Assistenzsystems zur rechnergestützten Analyse und Optimierung robuster Produkte*, vol. 356 of *VDI Fortschrittberichte, Reihe 20*. VDI-Verlag, Düsseldorf, 2002.

[72] HOCHMUTH, R., MEERKAMM, H., AND SCHWEIGER, W. An approach to a general view on tolerances in mechanical engineering. In *Proceedings of the 2nd International Workshop on Integrated Product Development, 1998, Magdeburg* (1998), pp. 65–76.

[73] HUANG, S. H., LIU, Q., AND MUSA, R. Tolerance-based process plan evaluation using Monte Carlo simulation. *International Journal of Production Research 42*, 23 (2004), 4871–4891.

[74] HUANG, Y. M., SHIAU, C. S., AND TAIPEI, T. A combined model for optimal tolerance allocation of assemblies. In *The 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2004, Albany, New York, AIAA 2004-4627* (2004), American Institute of Aeronautics and Astronautics.

[75] IILS – INGENIEURSGESELLSCHAFT FÜR INTELLIGENTE LÖSUNGEN UND SYSTEME MBH. DesignCompiler 43v2. Internet, http://www.iils.de/43.htm, accessed Jan 2012.

[76] JAYARAMAN, R., AND SRINIVASAN, V. Geometric tolerancing: 1. virtual boundary requirements. *IBM Journal of Research and Development 33*, 2 (1989), 90–104.

[77] JOHANNESSON, H. L. Computer aided analysis of functional couplings in structural axiomatic design. In *Proceedings of the Design Engineering Technical Conferences, 1995, DE-Vol. 82-1* (1995), American Society of Mechanical Engineers, pp. 337–343.

[78] JOHANNESSON, H. L. On the nature and consequences of functional couplings in axiomatic machine design. In *Proceedings of the ASME Design Engineering Technical Conferences and Computers in Engineering Conference, 1996, Irvine, California, 96DETC/DTM-1528* (1996), American Society of Mechanical Engineers.

[79] JOHANNESSON, H. L., AND SÖDERBERG, R. Structure and matrix models for tolerance analysis from configuration to detail design. *Research in Engineering Design 12*, 2 (2000), 112–125.

[80] KHODAYGAN, S., AND MOVAHHEDY, M. R. Tolerance analysis of assemblies with asymmetric tolerances by unified uncertainty-accumulation model based on fuzzy logic. *International Journal of Advanced Manufacturing Technology* (2011), 777–788.

[81] KLEIN, B. *Statistische Tolerierung*. Carl Hanser Verlag, München – Wien, 2002.

[82] KLEIN, B. *Toleranzmanagement im Maschinen- und Fahrzeugbau*. Oldenbourg Verlag, München – Wien, 2005.

[83] KLEMT, T., AND OLTMANN, K. M. Design optimization in aircraft component pre-design. In *Proceedings of SAWE 66th Annual International Conference on Mass Properties Engineering, 2007, Madrid, No. 3411, Category No. 10* (2007).

[84] KOLLER, F. *CAD-gestützte Toleranzrechnung basierend auf der Auswertung von Kontaktkräften in einem Mehrkörpersimulationsmodell*, vol. 219 of *VDI Fortschrittberichte, Reihe 20*. VDI-Verlag, Düsseldorf, 1996.

[85] KORMEIER, T. *Graphenbasierte Entwurfssprachen zur konsistenten Modellierung und musterbasierten Topologiemodifikation von Faserverbundstrukturen*. Dissertation, Universität Stuttgart, 2009.

[86] KORMEIER, T., AND RUDOLPH, S. Topological synthesis of shell structures. In *Proceedings of the ASME International Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE), 2006, Philadelphia, Pennsylvania, DETC2006-99092* (2006), American Society of Mechanical Engineers.

[87] KORMEIER, T., AND RUDOLPH, S. Rule-rased material topology modelling of composite structures. In *Proceedings of the 16th International Conference on Engineering Design (ICED07), 2007, Paris* (2007), pp. 515–516.

[88] KRAUSE, D., GEHM, M., AND HALFMANN, N. Visionary integration concepts for aircraft cabin interior. In *Proceedings of Workshop on Aviation System Technology (AST), 2009, Hamburg* (2009).

[89] KRIEGEL, J. M. Exact constraint design. In *Proceedings of the ASME International Mechanical Engineering Congress and Exhibition, 1994, Chigaco, Illinois, 94-WA/DE-18* (1994), American Society of Mechanical Engineers.

[90] KRÖPLIN, B., AND RUDOLPH, S. Entwurfsgrammatiken – ein Paradigmenwechsel? *Der Prüfingenieur (VPI) 26* (2005), 34–43.

[91] LA ROCCA, G. *Knowledge based engineering techniques to support aircraft design and optimization*. Dissertation, Delft University of Technology, 2011.

[92] LA ROCCA, G., AND VAN TOOREN, M. J. L. Knowledge based engineering to support aircraft multidisciplinary design and optimization. In *Proceedings of the 26th International Congress of the Aeronautical Sciences (ICAS), 2008* (2008).

[93] LANDES, B., AND RUDOLPH, S. Aircraft cabin architectures including tolerancing using a graph-based design language in UML. In *DGLR 60. Deutscher Luft- und Raumfahrtkongress (DLRK), 2011, Bremen, Tagungsband - Manuskripte (CD), No. 241303* (2011), pp. 691–697.

[94] LANTRIP, J., MUSKE, S., AND GLEADLE, J. GD&T for flexible contoured structures. In *Proceedings of the SAE/AIAA World Aviation Congress, 1997, Anaheim, California, No. 975604* (1997).

[95] LAUSCHER, T., FISCHER, C., AND HIEBENTHAL, T. High-quality interface specifications with SysML modelling. *Signal + Draht*, 3 (2011), 10–16.

[96] LEDERMANN, C. *Parametric associative CAE methods in preliminary aircraft design*. Dissertation, ETH Zürich, 2006.

[97] LEDERMANN, C., HANSKE, C., WENZEL, J., ERMANNI, P., AND KELM, R. Associative parametric CAE methods in the aircraft pre-design. *Aerospace Science and Technology 9* (2005), 641–651.

[98] LEE, D. J., AND THORNTON, A. C. The identification and use of key characteristics in the product development process. In *Proceedings of the ASME Design Engineering Technical Conferences and Computers in Engineering Conference, 1996, Irvine, California, 96-DETC/DTM-1506* (1996), American Society of Mechanical Engineers.

[99] LOSSACK, R. S. *Wissenschaftstheoretische Grundlagen für die rechnergestützte Konstruktion*. Springer-Verlag, Berlin – Heidelberg, 2006.

[100] LUSTIG, R. *Integration und Verarbeitung von Verformungsinformationen im Umfeld rechnerunterstüzter Toleranzanalysen*, vol. 418 of *VDI Fortschrittberichte, Reihe 20*. VDI-Verlag, Düsseldorf, 2008.

[101] MANARVI, I. A., AND JUSTER, N. P. Using FE to simulate the effect of tolerance on part deformation. In *Advances in concurrent engineering – Proceedings of the 9th ISPE International Conference on Concurrent Engineering, 2002, Cranfield* (Lisse, 2002), Swets & Zeitlinger, pp. 323–328.

[102] MANARVI, I. A., AND JUSTER, N. P. Framework of an integrated tolerance synthesis model and using FE simulation as a virtual tool for tolerance allocation in assembly design. *Journal of Materials Processing Technology 150* (2004), 182–193.

[103] MANNEWITZ, F. *Prozessfähige Tolerierung von Bauteilen und Baugruppen – ein Lösungsansatz zur Optimierung der Werkstattfertigung im Informationsverbund zwischen CAD und CAQ*, vol. 256 of *VDI Fortschrittberichte, Reihe 20*. VDI-Verlag, Düsseldorf, 1997.

[104] MANTRIPRAGADA, R., AND WHITNEY, D. E. The datum flow chain: a systematic approach to assembly design and modeling. *Research in Engineering Design 10* (1998), 150–165.

[105] MARGUET, B., AND MATHIEU, L. Tolerancing problems for aircraft industries. In *Proceedings of the 5th CIRP Seminar on Computer-Aided Tolerancing, 1997, Toronto* (1998), pp. 335–343.

[106] MARGUET, B., AND MATHIEU, L. Method for geometric variation management from key characteristics to specifications. In *Proceedings of the 7th CIRP Seminar on Computer-Aided Tolerancing, 2001, Cachan* (2003), pp. 217–226.

[107] MARWEDEL, S., FISCHER, N., AND SALZWEDEL, H. Improving the design quality of complex networked systems using a model-based approach. In *Proceedings of the 3rd International Conference on ModelŰBased Systems Engineering (ICŰMBSE2010), 2010, Fairfax, Virginia* (2010).

[108] MATHIEU, L., AND BALLU, A. GEOSPELLING: a common language for specification and verification to express method uncertainty. In *Proceedings of the 8th CIRP Seminar on Computer Aided Tolerancing, 2003, Charlotte, North Carolina* (2003).

[109] MAWHINNEY, P., PRICE, M. A., ARMSTRONG, C. G., CURRAN, R., EARLY, J., MURPHY, A., BENARD, E., AND RAGHUNATHAN, S. Design and analysis integration using systems engineering for aircraft structural design. In *4th AIAA Aviation Technology, Integration and Operations (ATIO) Forum, 2004, Chicago, Illinois, AIAA 2004-6204* (2004), American Institute of Aeronautics and Astronautics, pp. 20–29.

[110] MAWHINNEY, P., PRICE, M. A., ARMSTRONG, C. G., OU, H., MURPHY, A., GIBSON, A., AND CURRAN, R. Using idealised models to enable integration and analysis driven design. In *3rd AIAA Aviation Technology, Integration and Operations (ATIO) Forum, 2003, Denver, Colorado, AIAA 2003-6747* (2003), American Institute of Aeronautics and Astronautics, pp. 302–309.

[111] MAWHINNEY, P., PRICE, M. A., CURRAN, R., BENARD, E., MURPHY, A., AND RAGHUNATHAN, S. Geometry-based approach to analysis integration for aircraft conceptual design. In *5th AIAA Aviation Technology, Integration and Operations (ATIO) Forum, 2005, Washington, D.C., AIAA 2005-7481* (2005), American Institute of Aeronautics and Astronautics, pp. 1699–1707.

[112] MEADOWS, J. D. *Tolerance stack-up analysis: for plus and minus tolerancing, for geometric dimensioning and tolerancing*. James D. Meadows & Associates, Hendersonville, Tennessee, 2001.

[113] MECAMASTER SARL. MECAmaster user manual for MECAmaster version 7.2. Released with MECAmaster software distribution, MECAmaster SARL, 64 chemin des mouilles, 69134 Ecully CEDEX, France, 2010.

[114] MEERKAMM, H., AND HOCHMUTH, R. Integrated product development based on the design system MFK. In *Proceedings of the 5th International Design Conference (Design 98), 1998, Dubrovnik* (1998).

[115] MEHLITZ, P. Trust your model – verifying aerospace system models with Java pathfinder. In *Proceedings of IEEE Aerospace Conference, 2008, Big Sky, Montana* (2008).

[116] OLTMANN, K. M. Virtual engineering models for aircraft structure weight estimation. In *Proceedings of SAWE 66th Annual International Conference on Mass Properties Engineering, 2007, Madrid, No. 3418, Category No. 10* (2007).

[117] PAHL, G., BEITZ, W., FELDHUSEN, J., GROTE, K. H., WALLACE, K. (ED.), AND BLESSING, L. (ED.). *Engineering design: a systematic approach*. Springer-Verlag, London, 2007.

[118] PARK, G. J., LEE, T. H., LEE, K. H., AND HWANG, K. H. Robust design: an overview. *AIAA Journal 44*, 1 (2006), 181–191.

[119] PEAK, R. S., BURKHART, R. M., FRIEDENTHAL, S. A., WILSON, M. W., BAJAJ, M., AND KIM, I. Simulation-based design using SysML – part 1: a parametrics primer. In *Proccedings of INCOSE International Symposium, 2007, San Diego, California* (2007).

[120] PEAK, R. S., BURKHART, R. M., FRIEDENTHAL, S. A., WILSON, M. W., BAJAJ, M., AND KIM, I. Simulation-based design using SysML – part 2: celebrating diversity by example. In *Proccedings of INCOSE International Symposium, 2007, San Diego, California* (2007).

[121] PFAFF, J. M. *Parameterreduktion zur ähnlichkeitsmechanischen Gewichtsprognose im Flugzeugvorentwurf am Beispiel des Tragflügels*. Dissertation, Universität Stuttgart, 2008.

[122] PIMMLER, T. U., AND EPPINGER, S. D. Integration analysis of product decompositions. In *Proceedings of the ASME Design Technical Conferences, 1994, Minneapolis, Minnesota* (1994), American Society of Mechanical Engineers.

[123] POOLE, J. D. Model-driven architecture: vision, standards and emerging technologies. In *Proceedings of the 15th European Conference on Object-Oriented Programming ECOOP, 2001, Budapest* (2001).

[124] PRICE, M., EARLY, J. M., CURRAN, R., BENARD, E., AND RAGHUNATHAN, S. Identifying interfaces in engineering systems. *AIAA Journal 44*, 3 (2006), 529–540.

[125] PRICE, M., RAGHUNATHAN, S., AND CURRAN, R. An integrated systems engineering approach to aircraft design. *Progress in Aerospace Sciences 42* (2006), 331–376.

[126] PRISCO, U., AND GIORLEO, G. Overview of current CAT systems. *Integrated Computer-Aided Engineering 9* (2002), 373–387.

[127] RAYMER, D. P. *Aircraft design: a conceptual approach*. AIAA Education Series. American Institute of Aeronautics and Astronautics, 1999.

[128] REICHWEIN, A. *Application-specific UML profiles for multidisciplinary product data integration*. Dissertation, Universität Stuttgart, 2006.

[129] ROSKAM, J. *Airplane design, part I through VIII*. Design Analysis & Research Corporation, 1989.

[130] RUDOLF, H. *Wissensbasierte Montageplanung in der Digitalen Fabrik am Beispiel der Automobilindustrie*. Dissertation, Technische Universität München, 2006.

[131] RUDOLPH, S. *A Methodology for the Systematic Evaluation of Engineering Design Objects*. Dissertation (translation into english of the original german thesis), Universität Stuttgart, 1994.

[132] RUDOLPH, S. Upper and lower limits for 'The principles of design'. *Research in Engineering Design 8* (1996), 207–216.

[133] RUDOLPH, S. *Übertragung von Ähnlichkeitsbegriffen*. Habilitation thesis, Universität Stuttgart, 2002.

[134] RUDOLPH, S. Know-how reuse in the conceptual design phase of complex engineering products. In *Proceedings of Conference on Integrated Design and Manufacturing in Mechanical Engineering (IDMME), 2006, Grenoble* (2006), pp. 23–39.

[135] RUDOLPH, S. A semantic validation scheme for graph-based engineering design grammars. In *Design Computing and Cognition '06 – Proceedings of the 2nd International Conference on Design Computing and Cognition (DCC'06), 2006, Eindhoven* (Dordrecht, 2006), Springer, pp. 541–560.

[136] RUDOLPH, S. Digital Engineering – digitale Methoden für Entwurf und Fertigung komplexer Systeme. Lecture Notes, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 2011.

[137] SALOMONS, O. W., VAN HOUTEN, F., AND KALS, H. Current status of CAT systems. In *Proceedings of the 5th CIRP Seminar on Computer-Aided Tolerancing, 1997, Toronto* (1998), pp. 438–452.

[138] SCARR, A., JACKSON, D., AND MCMASTER, R. Product design for robotic and automated assembly. In *Proceedings of IEEE International Conference on Robotics and Automation, 1986* (1986), vol. 3, pp. 796–802.

[139] SCHAEFER, J., AND RUDOLPH, S. Satellite design by design grammars. *Aerospace Science and Technology 9* (2005), 81–91.

[140] SCHEER, A. W., WITTMANN, M., WEBER, C., AND THOME, O. Toleranz-Wissensbasis zur Unterstützung der integrierten Produktentwicklung. *VDI-Z 141*, 11/12 (1999), 18–20.

[141] SCHMIDT, L. C., AND CAGAN, J. Recursive annealing: a computational model for machine design. *Research in Engineering Design 7* (1995), 102–125.

[142] SCHMIDT, L. C., AND CAGAN, J. GGREADA: a graph grammar-based machine design algorithm. *Research in Engineering Design 9* (1997), 195–213.

[143] SCHOLZ, F. *Tolerance stack analysis methods*. Boeing Information & Support Services, Seattle, Washington, 1995.

[144] SCHOLZ, F. *Tolerance stack analysis methods – a critical review*. Boeing Information & Support Services, Seattle, Washington, 1995.

[145] SCHUT, E. J. *Conceptual design automation – abstraction complexity reduction by feasilisation and knowledge engineering*. Dissertation, Delft University of Technology, 2010.

[146] SCHUT, E. J., AND VAN TOOREN, M. J. L. Engineering primitives to reuse design process knowledge. In *Proccedings of 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2008, Schaumburg, Illinois, AIAA 2008-1804* (2008), American Institute of Aeronautics and Astronautics.

[147] SÖDERBERG, R. Robust design by tolerance allocation considering quality and manufacturing cost. *ASME Advances in Design Automation, DE-Vol. 69-1* (1994), 219–226.

[148] SÖDERBERG, R., AND JOHANNESSON, H. Tolerance chain detection by geometrical constraint based coupling analysis. *Journal of Engineering Design 10*, 1 (1999), 5–24.

[149] SÖDERBERG, R., AND LINDKVIST, L. Two-step procedure for robust design using CAT technology. In *Proceedings of the 6th CIRP Seminar on Computer-Aided Tolerancing, 1999, Twente* (1999), pp. 231–240.

[150] SÖDERBERG, R., AND LINDKVIST, L. Geometrical coupling analysis in assembly design. In *Proceedings of 2nd International Conference on Axiomatic Design ICAD, 2002, Cambridge, Massachusetts* (2002).

[151] SPOTTS, M. F. Allocation of tolerances to minimize cost of assembly. *Journal of Engineering for Industry 95*, 3 (1973), 762–764.

[152] SRINIVASAN, R. S., AND WOOD, K. L. Geometric tolerancing in mechanical design using fractal-based parameters. *Journal of Mechanical Design 117* (1995), 203–206.

[153] SRINIVASAN, V. ISO deliberates statistical tolerancing. In *Proceedings of the 5th CIRP Seminar on Computer-Aided Tolerancing, 1997, Toronto* (1998), pp. 25–35.

[154] SRINIVASAN, V., AND JAYARAMAN, R. Geometric tolerancing: 2. conditional tolerances. *IBM Journal of Research and Development 33*, 2 (1989), 105–124.

[155] STACHOWIAK, H. *Allgemeine Modelltheorie*. Springer-Verlag, Wien – New York, 1973.

[156] STEINBICHLER, G. *Methoden und Verfahren zur Optimierung der Bauteilentwicklung für die Spritzgießfertigung*. Dissertation, Universität Erlangen-Nürnberg, 2008.

[157] SUH, N. P. *The principles of design*. Oxford University Press, New York, 1990.

[158] SUH, N. P. Axiomatic design of mechanical systems. *Journal of Vibration and Acoustics 117* (1995).

[159] TORENBEEK, E. *Synthesis of subsonic airplane design*. Springer Netherland, 1982.

[160] TUDORACHE, T. *Employing ontologies for an improved development process in collaborative engineering*. Dissertation, TU Berlin, 2006.

[161] VAN TOOREN, M. J. L., LA ROCCA, G., KRAKERS, L., AND BEUKERS, A. Design and technology in aerospace – parametric modeling of complex structure systems including active components. In *Proccedings of the 13th International Conference on Composite Materials, 2003, San Diego, California* (2003).

[162] VAN WIE, M. J., GREER, J. L., CAMPBELL, M. I., STONE, R. B., AND WOOD, K. L. Interfaces and product architecture. In *Proceedings of the ASME International Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE), 2001, Pittsburgh, Pennsylvania, DETC01/DTM-21689* (2001), American Society of Mechanical Engineers.

[163] VERHAGEN, W. J. C., AND CURRAN, R. Knowledge-based engineering review: conceptual foundations and research issues. In *New world situation: new directions in concurrent engineering – Proceedings of the 17th ISPE International Conference on Concurrent Engineering (Advanced Concurrent Engineering), 2010, Cracow* (London – Dortrecht – Heidelberg – New York, 2011), Springer-Verlag, pp. 239–248.

[164] VOGEL, S., DANCKERT, B., AND RUDOLPH, S. Knowledge-based design of SCR systems using graph-based design languages. *MTZ 73* (2012), 50–56.

[165] WANG, Y. Semantic tolerance modeling. In *Proceedings of the ASME International Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE), 2006, Philadelphia, Pennsylvania, DETC2006/DAC-99609* (2006), American Society of Mechanical Engineers.

[166] WATSON, P., CURRAN, R., MURPHY, A., COWAN, S., HAWTHORNE, P., AND WATSON, N. A cost estimating model for aerospace procurement Pro-COST EST. In *4th AIAA Aviation Technology, Integration and Operations (ATIO) Forum, 2004, Chicago, Illinois, AIAA 2004-6237* (2004), American Institute of Aeronautics and Astronautics.

[167] WEILKIENS, T. *Systems Engineering mit SysML/UML – Modellierung, Analyse, Design*. dpunkt.verlag, 2006.

[168] WENZEL, J. Structural sizing for weight estimation in preliminary aircraft design. In *Proceedings of SAWE 66th Annual International Conference on Mass Properties Engineering, 2007, Madrid, No. 3421, Category No. 10* (2007).

[169] WHITNEY, D. E., GILBERT, O. L., AND JASTRZEBSKI, M. Representation of geometric variations using matrix transforms for statistical tolerance analysis in assemblies. *Research in Engineering Design 6* (1994), 191–210.

[170] WHITNEY, D. E., MANTRIPRAGADA, R., ADAMS, J. D., AND RHEE, S. Designing assemblies. *Research in Engineering Design 11* (1999), 229–253.

[171] WILHELM, R. G., AND LU, S. C. Y. *Computer methods for tolerance design*. World Scientific, Singapore, 1992.

[172] WITTMANN, M., AND SCHEER, A. W. *FIT: featurebasiertes integriertes Toleranzinformationssystem*, vol. 167 of *Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi) im Deutschen Forschungszentrum für Künstliche Intelligenz*. Institut für Wirtschaftsinformatik, Saarbrücken, 2000.