Integration of On-Board Power Systems Simulation in Conceptual Aircraft Design

Ingo Staack, Petter Krus
Division of Fluid and Mechatronics Systems
Linköping University
ingo.staack@liu.se

Keywords: Conceptual aircraft design, system simulation, onboard power systems, knowledge based engineering

Abstract
This paper describes the methodology of generating simulation models out of basic information, available during conceptual design phase. The implementation of an aircraft system is shown as an example using the simulation software HOPSAN.

Because of the limited direct project-related data available at the conceptual stage, the traditional method of creating physical simulation models by the bottom up approach with the help of (standard) component libraries is not applicable. Instead, the respective systems’ architecture as well as their composition has to be descriptively predefined in a flexible, wide-range applicable manner, known as the knowledge base (KB) approach. These system technology driven design declarations – combined with project related data – result in roughly pre-tuned system simulation models, which may help when conducting more detailed investigations of the project such as performance analysis.

This (system architecture) knowledge-based approach is shown on the whole aircraft system level down to the detailed implementation of the control surface actuator systems of the primary flight control system.

0 Nomenclature
In this paper, the following naming convention is used to describe a physical system (simulation model) composition:

model layout: the way in which a model consisting of (sub)systems or components is arranged
model topology: schematic description of the arrangement of a (sub)system, including its nodes, connecting lines and in-/output ports
model architecture: the structure and design of a model/system

1 Introduction
Classical aircraft conceptual design explained in the literature is mainly based on statistics of existing aircraft or basic physical effects (e.g. [1]). The natural limitation of these methods makes it necessary to perform more advanced, complex and more direct project property related analytical analysis, for example CFD and FEM if the product properties definition has reached a certain level of accuracy.

In a modern aircraft with its electrically driven subsystems, the entire systems become
closely integrated and both positive and negative cross-couple effects arise. This, combined with the enlarged system architecture complexity – owing to the higher overall efficiency – makes it almost impossible to investigate these effects without simulation. Taking into consideration all limiting factors and requirements might overload the engineer and require too detailed an input, which is not possible during the conceptual design phase where – apart from certification regulations – even central requirements are often vaguely formulated and negotiable.

A typical case is the primary flight control system (PFCS) with its control modes, which are normally not defined during conceptual design but are needed to some extent when simulating the total aircraft.

2 Related Work

This paper – with its focus on application – is related to a wide area of research within systems engineering and system architecture development and metamodelling.

Within model-based system engineering, the model creation and validation process is a major topic in addition to the system declaration. [2] shows the use of UML/SysML for system declaration within model-based development.

More focused on the requirement to product-function coupling, [3] defines a process that maps the requirements towards functions and the related means of the product (function-means tree) by generic object inheritance. The result can be seen as a configurable product data management (PDM) system that allows traceability to the requirements. [4] developed such a traceability (thus between models) with the help of the eXtensible Markup Language, XML [5]. Here, in the software engineering domain, related work focuses on reusable pattern definitions for the metamodelling process in XML, such as for example [6] with an overview of different pattern definitions.

Application examples of the whole process using UML for the graph-based metamodelling are given in [7] and [8].

3 Architectural Design Process

A simulation, together with model result investigations like stability and robustness analysis, helps the developer to understand the properties and behaviour of the design. Complex models, however, may veil or draw attention away from the important parameters; it requires a great amount of time to fine-tune all system parameters. Besides building the simulation model, system tuning (and if applicable system verification) is the most time-consuming and expertise-demanding process in system modelling. The target of the architectural design process is thus the generation of an already pre-tuned simulation model by knowledge reuse in combination with the project data.

3.1 Model Translation Methodology

Within the transition step from pure project data towards a simulation model, the combination of two processes is essential:

- **Transforming the project data** by an interpreter into the simulation system with its components and parameters.
- **Adding additional information** by pre-known knowledge such as for example the general architecture of a subsystem and the required components that are not explicitly described in the (conceptual aircraft) data setup.

![Figure 1: Flow diagram of the translation process.](image)

Figure 1 shows a comprehensive overview of this process where the project-related data (shown in red) are extended by the application
of more generally formulated application-related data (shown in green). The application-related data might for example be a KBS library for JAR/FAR-25 airplanes.

3.2 System Architecture Design Definition

The definition of the system and its layout can be seen as a metamodel of the simulation model, containing the necessary translation rules. These rules enable the design compiler to translate the aircraft project data in conjunction with the project-specific requirements and the general certification requirements\(^1\) into the executable simulation model.

Under closer scrutiny, the metamodel information can be split up into two parts:

1. the meta-data and the meta-process, dealing with the model (layout) generation
2. the data transfer processes in order to transfer the project data towards the (simulation) model systems and component properties.

3.3 KBS Level Definition

The system layout definitions used, in the following referred to as knowledge base system(s) (KBS), can be divided into four categories:

I. Fixed ports, static system

This represents a static system layout with fixed defined system ports. This type can be handled as a (complex) single component (KBC) but is matched towards a system. These types require only parameter adaption during instantiation, e.g. a simple propulsion system model, where only the engine deck data is updated.

II. Fixed Port, repetitive system

A fixed system component composition with an adaptable number of occurrences. An example is the mission system, defined by a state machine (with static number & functionality of in-/output ports) but with a flexible number of the repetitive elements of the same shape but different parameter settings.

III. Fixed port, flexible system

Systems with flexible system component composition but with a fixed defined number (and functionality) of the in-/output ports. An example is the PFCS controller with its roll/pitch/yaw commands (fixed ports!) but with significant changes in the controller layout between different projects (e.g. stable vs. unstable configuration).

IV. Flexible port, flexible system

Systems with flexible system component composition and varying number and functionality of the in-/output ports. Example is the hydraulic actuator system.

These KBS categories define and limit the possible actions within the meta-process.

Although automatic simulation model generation is the primary objective, this may not be applicable for every complex system (KBS category III or IV) definitions. In this case, the KBS translation may require user interaction, integrating the engineer (with his or her expertise) into the system architecture generating process [9]. This interaction can be defined by a configurator process; within the configurator, different design rules can be implemented to support the user during the designing process by preventing impossible combinations.

3.4 Simulation Compiling Process

Necessary inputs for the translation process, here called compiling, are:

- system topology definition (STD)\(^2\)
- KBS definitions (with the embedded configurator/adaption rules, mainly for KBS categories III and IV)
- KBE definitions establishing the connection to the simulation component libraries used

---

\(^1\) In the example implementation in this paper, certification requirements have not been included.

\(^2\) The STD might be integrated in the system’s KBS.
• the project-related dataset (in a known setup)

The process itself is then:
• instantiate the KBS
• conduct the configurator process (if applicable)
• apply the (project data) adaption rules
• create the inter-KBS connections, defined by the STD.

These steps can be conducted iteratively top-down on the different subsystem levels.

3.5 Linkage and Coupling Adaptation

With the presence of type IV KBS, the overlying system description has to be capable to establish valid connections according to both the KBS definition and the simulation program’s needs. For this purpose, it might be necessary to define one’s own “connection” KBS that are added by the STD.

In the case of HOPSAN [10], this is especially tricky because of its underlying TLM solver\(^3\) and the fact that most connector ports of the standard library components are not multi-connection capable. In the latter, the KBS has to include the process information on how to merge these ports. This merging process can be solved either by adding only a KBC with the required number of ports or a multiport functionality, or by adding one’s own subsystem (of KBS type IV). A similar approach can be used for split-up processes.

In the aircraft example, these merging processes are used for the distribution (and signal gain adaption) of the roll, pitch and yaw PFCS commands for the control surface actuators.

3.6 Simulation Parameter Tuning

In addition to the system layout modelling process, the respective component properties have to be set. These may be fairly easy parameter aliases in high-level systems or components that directly match a subset in the existing (conceptual design) project data; this is the case in the 6DOF aircraft model, where the aerodynamic coefficients, geometry and weight properties of the simulation model (which is a single component defined by the related KBC definition) are directly linked with the project data. This process can be seen as a simple data translation and can thereby easily be performed by for example an XSLT [11] file.

This topic becomes more complex on the detailed subsystems and components that are not explicitly or implicitly defined in the project data. This is the case where new information is added to the project using the knowledge base approach, for example the respective actuators and the mechanic actuator – control surface linkage: This can be solved by a complex approach making use of the (system, design space and requirements) limitations, the model (analysis result) data and the KBS definitions. In the case of actuator and linkage sizing these are:

• available space for the actuator and linkage (defined by the wing and airfoil shape) \([\text{geometrical scaling}]\).
• control surface hinge moments and forces from the aerodynamic analysis \([\text{power scaling}]\), together with:
• (general knowledge of) usual control surface deflection and deflection angle \([\text{geometry & power scaling}]\)

Iteratively, the connected parent systems can be built up and tuned in the same manner (bottom-up approach) like e.g. the hydraulic power supply system scaling according to the flow requirements of the connected actuators.

---

\(^3\) The simulation software HOPSAN is based on a distributed system solver method (TLM) with two different component type definitions which have to be alternating connected. For more information see [12].
3.7 Model tuning by Simulation

This method has not been applied to the project yet, but has to be named because of its importance and powerful capabilities:

Because the outcome of the process shown above is an executable simulation, the retrieved simulation results can be used to apply a simulation based optimization on the generated system, not primarily focused on a real system optimization but in order to retrieve a well-functioning (and thereby hopefully well-tuned) simulation model.

4 Use Case: Total Aircraft and Hydraulic System Simulation

For this paper, a use case has been chosen to demonstrate the implementation of the methodology shown above: a total aircraft system simulation with the primary flight control (hydraulic power) system, based on conceptual aircraft data.

4.1 Aircraft System Topology Definition

A flexible aircraft system top level architecture can be defined as shown in Figure 2.

![Figure 2: Top level aircraft system description.](image)

This architecture is a logical composition of the physical instances needed to describe a whole aircraft.

4.2 Aircraft Hydraulic System Topology

The (normally hydraulic) actuator power system of the PFCS can be described by three different system layouts (see Figure 3):

- Centralized system
- Distributed system
- Hybrid systems [12]

![Figure 3: Hydraulic system top level architecture layout concepts.](image)

The naming refers to the supply power generation concept. In civil applications, the centralized layout is usually realized with the trend towards hybrid systems in the latest aircraft generations, whereas in military applications both centralized and distributed systems can be found (see Figure 3).

4.3 Centralized Hydraulic System Definition

The following application example will focus only on the centralized system layout.

![Figure 4: Airbus A-320 Hydraulic system layout [13].](image)
A typical system in a civil JAR/FAR-25 certified aircraft, the Airbus A-320/321 family, is shown in Figure 4 [13]. These types of systems usually consist of three independent hydraulic systems, here called yellow, green and blue.

As described in Chapter 3.2, the system-designing process has to be described in STD/KBS declarations. Here, XSD, an XML schema language notation, has been used [14] for these definitions. This offers the potential to use the file during the whole process; for the system architecture (rules) development, the documentation and the instruction source for the design compiler.

The definition (of the PFCS related parts) of one of the centralized hydraulic systems in the Airbus A-320 shown above can be described as shown in Figure 5, where each element beginning with the prefix KBS refers to a system layout definition devised by the designer. The KBC prefix denotes elements that are directly linked towards simulation components and thus represent the smallest possible unit that cannot contain any subsequent KBS/KBE definitions.

4.4 Hydraulic Actuator Model Definition

By defining the global system layout as “centralized”, the actuator type used can be only of hydraulic input type. Figure 6 shows a hydraulic actuator configuration tree, which can be easily simplified for the PFCS case to:

- linear type only

- civil: usually single or double type
- military: usually tandem type [15]

The configured actuator therefore encapsulates no further KBS definitions (KBC only), this system can be instantiated by the compiler process. In the case of the following selection

```typescript
hydraulicActuator(type="linear" housing="single" subtype="unbalanced");
```

this declaration is translated into the following actuator subsystem:

![Actuator subsystem](image)

Figure 6: Hydraulic Actuator design tree. Double and triple linear actuator can be seen as a combination of single actuators; rotary type branch not shown.

For the overlying KBS, this object appears as a black box with the defined system in-/output ports visible as shown in Figure 8.

![KBS instantiation](image)

Figure 7: Example of the KBS instantiation of a linear single actuator (KBS Type I) in HOPSAN.

![Top level view](image)

Figure 8: Top level view of the linear actuator system (without defined system symbol) with the highlighted system ports hydraulic (green), mechanical (blue) and control signal input (red).
The actuator KBS therefore contains no further KBSs. The created simulation object code, shown in Figure 9, consists only of library components with their parameters and connections and the system ports.

5 Model-Framework Implementation

In this example, Matlab scripts have been used as the design interpreter, with the following inputs:

- TANGO aircraft data (Matlab class or XML).
- the KBS (including the STD), implemented in Matlab.
- Configurator with GUI (also Matlab).

5.1 Conceptual Aircraft Design Framework

The aircraft data has been generated with the conceptual aircraft design tool TANGO, developed at the Division of Fluid and Mechatronic Systems (FluMeS) at Linköping University [16]. This Matlab application saves the aircraft sizing output in a parametric-based datasheet in XML format, similar to the CPACS definition from DLR [17].

Together with the panel method program Tornado [18] included in TANGO and the CATIA based RAPID program these tools form a powerful framework for conceptual aircraft design [16].

5.2 HOPSAN Simulation Environment

The in-house developed multi-domain system simulation tool HOPSAN [10] was used for the simulation.

HOPSAN works with a distributed, fixed time step solver for each component, using the transition line method [19]. Using the HOPSAN standard library, a six degree of freedom aircraft simulation model, including mission controller, propulsion and the PFCS, can be simulated [20]. On a standard PC, simulation time is approximately 30 seconds for a whole mission.
5.3 The Model Generation Process: Model Description and Translating Action

The project-related model data, i.e. the aerodynamic properties, are achieved during the conceptual aircraft design with help of a vortex lattice method program, Tornado [18], and classical textbook methods. By means of these parameters, a six degree of freedom (6DOF) aircraft model can be created and placed into the simulation as the central part in the highest system layer. More details about the 6DOF model and the total aircraft system simulation can be found in [20].

Executing the Matlab design interpreter script starts the KBS defined system with the top layer architecture according to Figure 2. During the compilation process, the user is integrated in the hydraulic system layout configuration through configurator GUIs.

![Figure 12: The generated HOPSAN simulation model (top level view).](image)

Components are placed within the subsystems using a simple algorithm that is not capable of fulfilling the need for good, human readable component placement and connector line shape. Figure 13 shows the manually realigned hydraulic system with its colour-coded connections and the connection subsystems created (see in Chapter 3.5) which serve to merge the actuator position outputs to the control surfaces.

![Figure 13: The generated centralized hydraulic system with three hydraulic systems; comparable with Figure 4 (components manually rearranged).](image)

5.4 Simulation Results, Analysis and Representation

In addition to the HOPSAN GUI and the simulation result analysis, the simulation model (setup) can be directly analysed during instantiation: Figure 14 shows different model analysis and access methods that are available to the developer during the simulation model generation process. These are (from left to right) the Matlab class(es) instance of the model, a component connection analysis (connection matrix) and a tree view of the subsystems and component hierarchy. It should be noted that this hierarchy differs from the original (TANGO XML style sheet) dataset hierarchy. In order to fit industrial (documentation) standards, this analysis data can also be exported in for example Microsoft Excel or HTML format.

![Figure 14: Overview of different simulation representation and analysis in Matlab, Microsoft Excel or XML-based mind maps.](image)

5.5 Limitation and Use Case Review

The use case implementation shown was performed with a rather crude implementation in Matlab with a focus on testing the suggested
methodology and developing the KBS/KBE/SDK nomenclature. The following
problem areas were detected during this work:

- graphical representation of the generated system; requires complex component placing algorithm.
- insufficient KBE definition assistance for the developer: Other program languages and development environments should be tested for the KBS/KBE definition.
- aircraft system layout is mainly driven by reliability. This makes it necessary to involving fault tree analysis (FTA) and failure mode and effect analysis (FMEA) during the KBS definition process.
- Time-consuming KBS definition; only justifiable if the KBS definitions are reusable in other/future projects or are integrated in an iterative (optimization) process.

6 Conclusions
Simulation within aircraft conceptual design is – to some extent – state-of-the-art. This paper shows that with additional effort, even complex and detailed simulation models can be created by applying knowledge-based engineering methods. Primarily this is shown as a support tool for the engineer, but the (automatic) breakdown of systems in particular enables a more accurate system efficiency and weight prediction and will thereby enhance the project benchmark prediction accuracy.

In the PFCS use case shown above, using a robust default (knowledge-based) topology for these (kinds of) systems might not get the full capability the aircraft will attain later but gives the engineer a chance to evaluate the aircraft.

A key prerequisite for this approach is a parametric, well-defined (input) dataset that allows parsing of the original data into simulation component properties or whole system architectures. The XML format in combination with its related style and translation languages (XSLT and XSD, both XML-based) suits very well; theoretically, the translation might be performed directly by a XML Transformation Sheet (XSLT), but initial proves highlighted the limitations of this language regarding mathemathic operations and complex parsing actions. The KBE description and design compiling were performed by means of Matlab scripts instead. Future work will include replacing these scripts with an appropriate language (e.g. SysML, UML or XSD) and specifying the KBS nomenclature.

References
[5] XML 1.0 Language Definition http://www.w3.org/XML (2013.08.17)

---

4 using the XSLT Version 1.0 in combination with XPath Version 1.0; Currently existing XSLT Versions 2.0 (and Version 3.0 announced) may offer greater functionality.
Integration of On-Board Power Systems Simulation in Conceptual Aircraft Design

Aircraft System Technologies (AST), TUHH, Hamburg, Germany, 2013, pp. 159–168.


