INTEGRATED FUNCTION MODELLING: COMPARING THE IFM FRAMEWORK WITH SYSML

Eisenbart, Boris (1); Mandel, Constantin (2); Gericke, Kilian (2); Blessing, Lucienne (2)
1: University of Sydney, Australia; 2: University of Luxembourg, Luxembourg

Abstract

This paper presents a comparison between the Integrated Function Modelling framework and SysML with the aim of deriving specific potentials for cross-fertilisation and further improvement regarding their application for function modelling in interdisciplinary design. The presented comparison comprises literature reviews as well as the practical application of both the IFM framework and SysML for modelling the functionality of an exemplary mechatronic system. The research leads to the identification of advantages and shortcomings in both approaches. Based on these insights, the paper further presents a conceptual adaptation of the IFM framework with the intention to improve its practical applicability and reducing modelling efforts.

Keywords: Functional modelling, Conceptual design, Integrated product development, Product modelling

Contact:
Dr. Boris Eisenbart
University of Sydney
Discipline of International Business
Australia
boris.eisenbart@sydney.edu.au

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 20th International Conference on Engineering Design (ICED15), Vol. nn: Title of Volume, Milan, Italy, 27.-30.07.2015

1 INTRODUCTION

The development of multi-technology systems requires designers from various disciplines to collaborate closely and establish a shared understanding of the problem and the emerging solution alike (Kleinsmann and Valkenburg 2008). This entails clarification of the requirements, central expected functions and their dependencies, as well as elaboration of different solution elements and their implementation (Frankenberger et al. 1998). Function modelling contributes to the development of a shared understanding in the design team as it addresses solution finding early in the process and on an abstract level (Chakrabarti and Bligh 2001). In particular, the combination of function modelling with (initial) modelling of a system's structure is suggested to help with building this shared understanding (Shai and Reich 2004, Eisenbart 2014). However, a large variety of alternative function modelling approaches exist and are frequently incompatible as they address divergent contents and are based on varying schemes for reasoning about functions and potential solution concepts (Erden et al. 2008, Eisenbart et al. 2012, 2013a, 2013b). While different scholars advocate establishing a shared function model that is used *in addition* to models used within disciplines to address this issue, others – including the authors of this paper – argue that providing *an integrative model* which aggregates disciplinary function modelling is of greater benefit. An integrative model allows designers from different disciplines and design contexts to relate between information relevant to them and others, which is expected to contribute to the development of a holistic, shared understanding of a system's functionality. Two approaches that integrate particularly many relevant contents from function modelling across disciplines and combine them with a representation of the system structure are the Object Management Group Systems Modeling Language (OMG 2012, hereafter "SysML") and the recently proposed Integrated Function Modelling (IFM) framework (Eisenbart et al. 2013c). The authors consider both amongst the most promising approaches for integrative function modelling and they are therefore further investigated in this paper. They differ fundamentally in how information is modelled: while SysML is a strongly formalised modelling language, the IFM framework is a flexible approach for representing and visually linking information in a matrix format. The presented research aims to compare both approaches and derive potentials for cross-fertilisation in order to combine their respective strengths.
Sections 2 to 4 introduce both modelling approaches briefly and provide an initial comparison based on a review of relevant literature. Section 5 illustrates the application of both approaches by modelling an exemplary mechatronic system. The findings from literature review and practical application are consolidated in Section 5 to derive specific potentials for cross-fertilisation. Based on these insights, a conceptual adaptation of the IFM framework is proposed to improve its applicability (see Section 6).

2 THE INTEGRATED FUNCTION MODELLING FRAMEWORK

The IFM framework is intended to provide designers with an integrated, cross-disciplinary approach for modelling system functionality. Following Eisenbart (2014), functions are defined as an intended or already perceivable behaviour of a technical system intended to fulfil a task. In the IFM framework integration is facilitated through linking the specific contents prominently addressed within discipline-specific function models (i.e. in abstract behavioural modelling) and is further complemented with initial system structural modelling. The IFM framework and its application are described in detail in Eisenbart et al. (2013c, 2014) and Eisenbart (2014). It consists of associated views as illustrated in Figure 1.

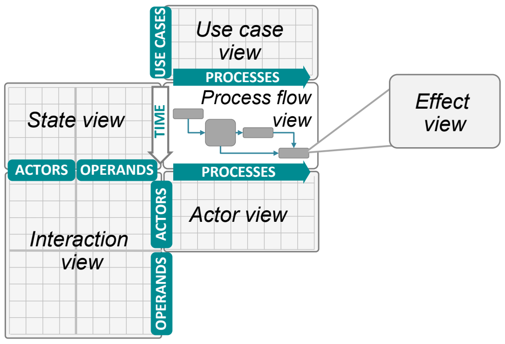


Figure 1. The IFM framework

A central view (*process flow view*) represents the flow of transformation and interaction processes, which are central in function modelling irrespective of disciplines (Eisenbart et al. 2013a). The remaining views are linked to this central view and comprise of matrices representing information about the different entities in the framework and their interdependencies equivalent to Multi-Domain-Matrices (MDM, see Kreimeyer and Lindemann 2011). Inherent entities centrally comprise of *use cases*, *transformation* and *interaction processes*, *effects*, *states*, *operands* and *actors*. Use cases represent different scenarios of applying the technical system for a specific purpose (e.g. fulfilling a goal, changing the state of the system or user, etc.). Transformation processes describe *technical* and/or *human* processes – realised by basic physiochemical effects – that result in a change of state of operands or actors. Operands are specifications of energy, material and signals. Actors comprise stakeholders (referring to any human or other animate being), technical (sub-)systems (which may be hardware and/or software) and parts of the environment that are actively or passively contributing to function fulfilment. Finally, interaction processes describe "cross-boundary" interactions between different actors jointly contributing to function fulfilment (see also Eder and Hosnedl 2008). The different views are briefly described in Table 1. Their adjacent placement (see Figure 1) supports their parallel development and allows verification of their mutual consistency across the entire framework. Furthermore, views are modular and may be added or omitted in order to allow for flexible adaptation of the framework to the specific demands of practitioners (resulting from diverse working sequences and rationales of different disciplines in different companies). This particular setup enables different disciplines to work flexibly with the individual views in different kinds of design projects (i.e. original, variant or adaptive design) and contribute to their iterative development during the solution finding process.

Table 1. Associated views in the IFM framework

View	Description
Process flow view	…qualitatively visualises the flow of sequential or parallel (interaction and/or transformation) processes related to a specific use case. For each use case an associated set of views is created. In the vertical direction, the process flow is visualised related to time. This matches to the flow of states in the associated *state view*. In horizontal direction, the process blocks are spread from left to right to enable a direct link to the *actor view* (see Figure 2).
State view	…represents the states from initial to final of operands and actors as well as their changes associated to the flow of individual processes. It can also be indicated if an operand or actor merely support a process without changing their own state.
Actor view	…indicates the involvement of one or more actors in the realisation of individual processes related to a use case. Involvement may be active or passive. Actors can be differentiated based on whether they – from the designers' point of view – are part of the system or not, which further separates *transformation* from *interaction* processes (realised "cross-boundary").
Use case view	…indicates the involvement of individual processes in the different use cases. It is intended to support analysis of dependencies between processes, which may hinder their parallel or sequential execution and thus the operability of use cases in which they are involved.
Effect view	…represents the effects, which enable individual processes and are provided by actors. For each process block in the *process flow view*, a separate *effect view* may be created, preferably using a similar representation. This allows for detailed analysis of the basic physiochemical effects that are affecting or contributing to the individual processes.
Interaction view	…uses a combination of matrices which map the specific bilateral impacts between actors and operands as well as their complementary contributions (or any other kind of dependency between them) in the realisation of use cases, associated processes etc. Additionally, information about the embodiment of specific bilateral impacts may be included. Hence, this view essentially results in an initial system structure or interface matrix of the system, respectively.

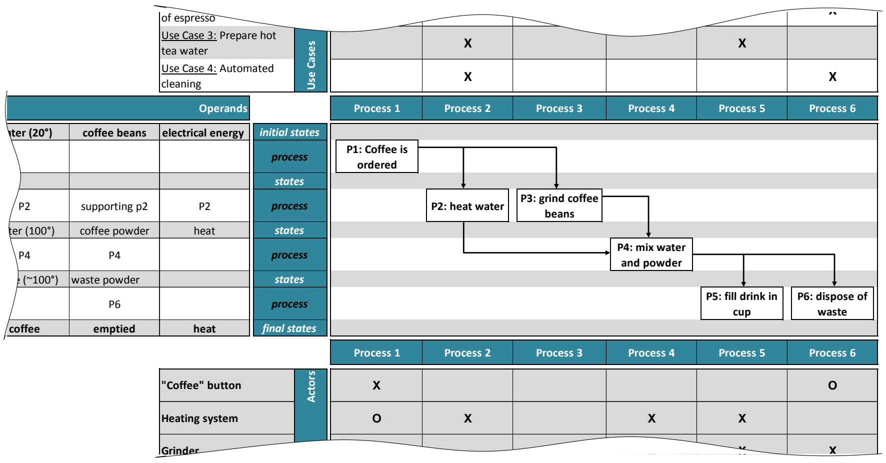


Figure 2. Example of a process flow view *(centre) for use case "prepare a cup of coffee" with adjacent* use case view *(upwards),* actor view *(downwards) and* state view *(left), adapted from Eisenbart (2014)*

3 SYSML

SysML is intended to offer a consistent and formalised way for representing (basically any kind of) information. The used formalism aims to improve clarity to the model users, thus – eventually – supporting model comprehension and communication between designers irrespective of their disciplinary background. Succeeding from UML, SysML is a so-called "visual modelling language" (Friedenthal et al. 2006). As such, it is generic in terms of what can be represented with it; however, a central set of models (typically referred to as "diagrams") has been established over the years (see e.g. Weilkiens 2008). These allow for detailed modelling and analysis of requirements as well as a system's structure and behaviour. Extensions to these diagrams are frequently discussed in literature to allow customisation to specific demands of the designers. The analysis presented here focuses mainly on the central diagrams proposed for abstract behavioural modelling (i.e. function modelling, as indicated above), which are briefly described in Table 2 (see Weilkiens 2008, Friedenthal et al. 2006).

A variety of commercial and open-source software environments are available to support modelling using SysML. Information about entities modelled in the different partial models (i.e. the different diagrams) is typically stored in a meta-model serving as a database for the modelling process. While individual diagrams are separated from each other, the entities they represent can be re-used from the database. Hence, each diagram represents specific information, while they complementarily illustrate all available information about the system. Entities need to be defined once using a *block definition diagram* (see Table 2) to start with.

Table 2. Central diagrams for function modelling proposed by Weilkiens (2008)

Diagram	Description
Block definition diagrams	…are used for definition purposes and modelling of relations between different entities. Any kind of entity may be modelled using a new block. Each block can be equipped with attributes (i.e. physical properties), operations (i.e. technical processes or activities), values, etc. These diagrams thus specify any relation between entities, including their mutual impacts and dependencies. Typically, this concerns modelling a system under development, any surrounding human being (such as users, referred to as "actors" in SysML) or peripheral technical systems. Each block can be further specified with a so-called *internal block diagram*. These may be used to detail the connections and parts inside the system as well as their usage (see e.g. Friedenthal et al. 2006).
Use case diagrams	…represent the use cases a system is associated with as well as different users, peripheral technical systems and their involvement within the use cases. Mutual connections among use cases (e.g. one use case including several others) may also be represented.
Activity diagrams	…represent the flow of processes performed by users, peripheral technical systems or the system to be developed (and/or its components) during the execution of a use case. Activity diagrams include all sequential, parallel and alternative processes, error scenarios, etc. to fully describe the devolution of a use case.
Sequence diagrams	…represent the interactions between all users, peripheral technical systems and the system to be developed during the devolution of a use case. Alternatively they can be used to represent individual, more complex processes, in detail. The involved users and technical systems are represented by so-called "lifelines". They exchange "messages" and interact with the system as well as among each other. This essentially corresponds to a flow of operands (mostly information). Interactions are modelled chronologically from top to bottom. Interactions may evoke subsequent processes to take place, which can be modelled using an additional *sequence diagram*.
State machine diagrams	…are associated to one specific block in an (internal) block diagram. They describe the entity's states and their changes during its life cycle. State changes are usually evoked by processes. These are referenced by a so-called "trigger" in the *state machine diagram*.

Weilkiens (2008) proposes a sequence of modelling steps to be applied while generating the different diagrams. For abstract behavioural modelling, central use cases in the system's life-cycle and the main involved actors are initially determined and represented in a *use case diagram*. The devolution of these main use cases can be modelled in an *activity diagram*, which illustrates the sequence in which the individual use cases may be executed in the different phases of the system's life-cycle. Subsequently, for each of these use cases, the specific sequence of required processes is modelled in a separate *activity diagram*. In parallel, associated *sequence diagrams* may be generated to provide more detail. Finally, modelling the process flows for each use case may be further substantiated with *state machine diagrams*. Associated *block diagrams* can be used to represent the system's structure. The different diagrams are to be iteratively refined.

4 INITIAL COMPARISON

IFM framework and SysML are initially compared based on the descriptions and publications cited in the previous sections, as well as insights from on-going research on the IFM framework by the authors of this paper and existing research on SysML including – but not limited to – Peak et al. (2009), Bone

and Cloutier (2010), and Pires et al. (2012). Here, the selected criteria for the comparison mainly focus on conceptual differences between the approaches that affect the actual application (see Section 5).
To start with, differences were found regarding the addressed entities and their respective definitions. Considering function modelling, a fairly similar set of entities compared to the IFM framework is addressed in the standard diagrams proposed for SysML. The identified differences concern *effects*, which are usually absent in the standard diagrams introduced earlier, and *stakeholders*. Stakeholders in SysML involve external (groups of) human beings that have a general interest in the system under development, without necessarily taking part in function fulfilment. In addition, while in the IFM framework human beings explicitly can be part of the system under consideration (e.g. in a Product-Service System), Weilkiens (2008) focuses on technical systems. Humans (e.g. users) are essentially considered interacting elements of the environment. In the following discussions, the terminology proposed for the IFM framework is used for both approaches to have a common basis.
Concerning the modelling, a central difference is the strong formalism in the diagrams and their application in SysML in contrast to the large degree of freedom in applying the IFM framework. Further aspects discussed in the following include the specific *area of application*, *software tool dependency*, realisation of the *linkage between different diagrams/views* in the approaches, *change management*, as well as *model and work partitioning*. The findings are collocated in Table 3.

Table 3. Initial comparison between SysML and IFM framework

SysML	IFM framework
Area of application	
• Can support the whole design process, starting with requirements modelling over function modelling to structural (and parametrical) modelling; • Further provides links to typical system simulation tools used during detail design (thus wider in scope as the IFM framework).	• Particularly focuses on modelling and analysing system functionality consistently coupled with initial system structural modelling (in the *interaction view*); • May be coupled with – but does not include itself – requirements modelling and early system simulation (see Dohr et al. 2014).
Software tool dependency	
• In principle, diagrams may be created using any kind of graphic modelling tool (e.g. MS PowerPoint, Visio, etc.); for complex systems, use of an appropriate software tool is crucial: they provide means to reuse entities from the meta-model in different diagrams (as discussed in the previous section). • Software tools can support SysML formalism to be maintained.	• Thus far, no software modelling tool for IFM framework existent; modelling may be performed using spreadsheet tools like MS Excel, while for less complex systems, matrices/views may be sketched by hand; • However, on-going research shows that implementation into a software tool can be well performed (see Dohr et al. 2014).
Linkage between different diagrams/views	
• Uses a meta-model as database storing all information about entities and their relations (as described before), individual diagrams are used to represent specific information graphically; • Entities are allocated from meta-model into the diagrams, thus the meta-model provides the (logical) linkage between model elements represented in different diagrams.	• The IFM framework particularly emphasises direct visual linkage of associated and adjacently placed views; • Inherent matrices share header rows and header columns, in order to facilitate traceability and linkage between contents of views (foreseen to facilitate analysis of their mutual consistency and ease gradual refinement/adaptation during the design process).
Change management	
• (Provided, entities are correctly defined/allocated from the meta-model into the diagrams) adaptations concerning same entities in one diagram, are automatically advanced to others; yet, entirety of caused effects (i.e. whether all affected entity relations will remain consistent) hardly verifiable (especially, in case diagrams partitioned to multiple designers/departments); • Change automatism is provided by – and thus depends on – the specific implementation in the used software tools.	• Effects of changes made to entities and their relations in individual views can be directly visually traced by designers for each set of views through verification of consistency of associated rows/columns in adjacently placed views (see particularly Eisenbart et al. 2014).
Model and work partitioning	
• Possibilities to partition the information to be modelled by different designers/departments with access right management (i.e. managing who has the right to see and change which specific information in the meta-model)	• Intended for collaborative modelling; • Modelled information explicitly separated per use case, i.e. each set of associated views focuses on one use case only.

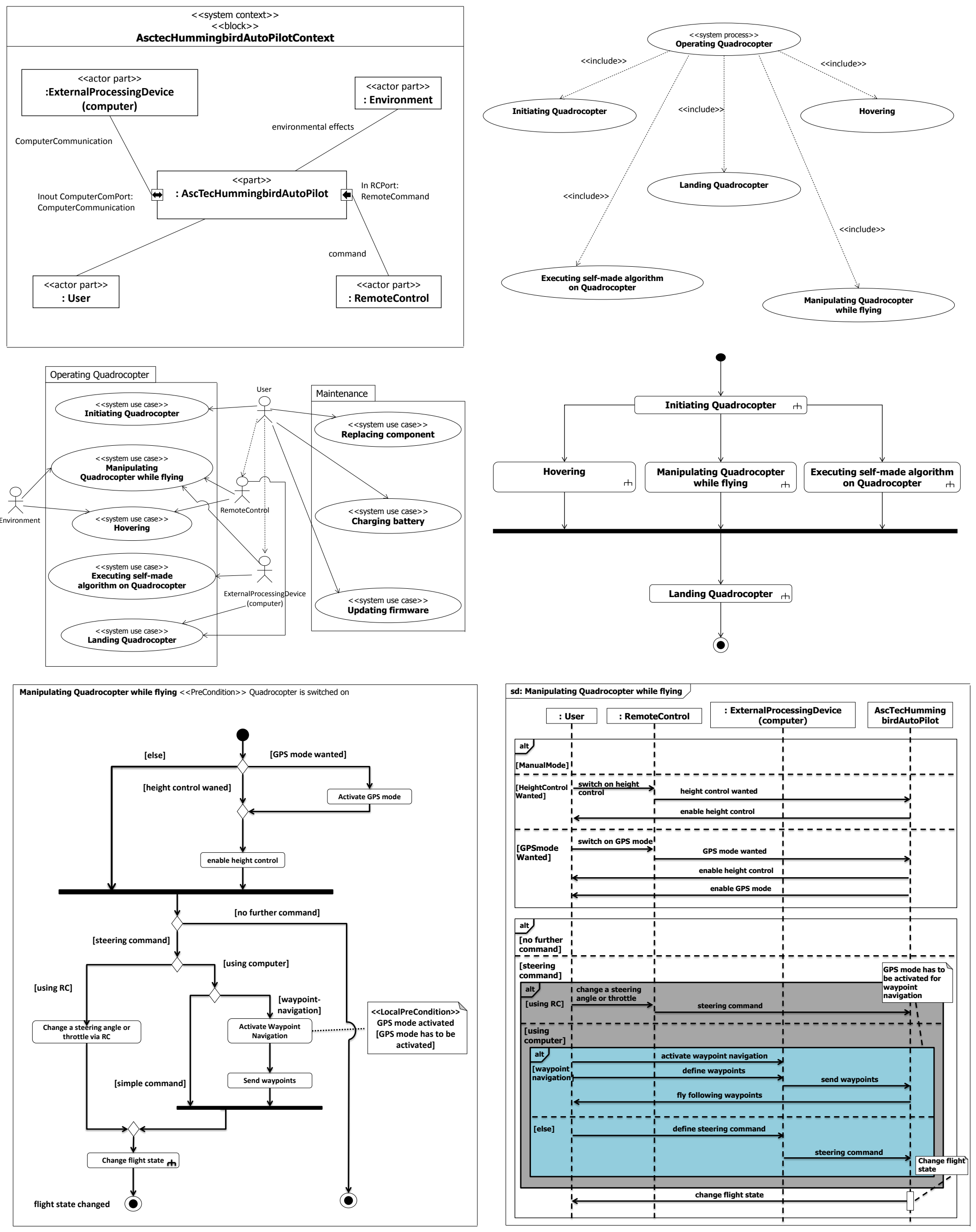
5 MODELLING AN EXEMPLARY MECHATRONIC SYSTEM

In this section the initial comparison is advanced based on the analysis of two models of a mechatronic system's functionality and initial system structure as well as the experiences gained during their generation by the authors. One model is based on the IFM framework, while the other is based on the SysML formalism using the diagrams described in Table 2. The comparison focuses on the practical applicability of both approaches (that result from the discussed conceptual differences), in order to derive concrete potentials for cross-fertilisation and improvement, as the central aim of this research.

5.1 Function modelling of a quadrocopter

The selected example is the AscTec Hummingbird, a self-stabilising, unpiloted aerial vehicle (a quadrocopter) that can be controlled using a remote control or a computer. The Hummingbird is considered suitable for the comparison as it provides diverse functionality, supports several use cases, and combines various engineering technologies. Seven main use cases can be discerned: *initiating the quadrocopter*, *manipulating quadrocopter while flying*, *hovering*, *landing quadrocopter*, *replacing components*, *charging battery*, *updating firmware*.

Each of these use cases was initially modelled with SysML (using Eclipse/Papyrus). Modelling was mainly performed by a bachelor student in his final year with a background in mechanical engineering and mechatronics in collaboration with the remaining authors of this paper. Figure 3 exemplarily shows a selection of diagrams for modelling the use case *manipulating quadrocopter while flying* in SysML. It includes a few variant process flows in relation to the process of steering: the quadrocopter can be remotely controlled by a person, by a computer, by waypoint navigation, with or without GPS support. Variants in Figure 3 are represented as alternative paths in *activity* and *sequence diagrams*.



<<system context>>
<<block>>
AsctecHummingbirdAutoPilotContext
<<actor part>> :ExternalProcessingDevice (computer)
<<actor part>> : Environment
ComputerCommunication
environmental effects
<<part>> : AscTecHummingbirdAutoPilot
Inout ComputerComPort: ComputerCommunication
In RCPort: RemoteCommand
command
<<actor part>> : User
<<actor part>> : RemoteControl
<<system process>> Operating Quadrocopter
<<include>>
Initiating Quadrocopter
Hovering
Landing Quadrocopter
Executing self-made algorithm on Quadrocopter
Manipulating Quadrocopter while flying
Operating Quadrocopter
Maintenance
User
Environment
RemoteControl
ExternalProcessingDevice (computer)
<<system use case>> Initiating Quadrocopter
<<system use case>> Manipulating Quadrocopter while flying
<<system use case>> Hovering
<<system use case>> Executing self-made algorithm on Quadrocopter
<<system use case>> Landing Quadrocopter
<<system use case>> Replacing component
<<system use case>> Charging battery
<<system use case>> Updating firmware
Initiating Quadrocopter
Hovering
Manipulating Quadrocopter while flying
Executing self-made algorithm on Quadrocopter
Landing Quadrocopter
Manipulating Quadrocopter while flying <<PreCondition>> Quadrocopter is switched on
[else]
[GPS mode wanted]
[height control waned]
Activate GPS mode
enable height control
[no further command]
[steering command]
[using computer]
[using RC]
[waypoint-navigation]
Change a steering angle or throttle via RC
Activate Waypoint Navigation
<<LocalPreCondition>> GPS mode activated [GPS mode has to be activated]
Send waypoints
[simple command]
Change flight state
flight state changed
sd: Manipulating Quadrocopter while flying
: User
: RemoteControl
: ExternalProcessingDevice (computer)
AscTecHummingbirdAutoPilot
alt
[ManualMode]
[HeightControl Wanted]
switch on height control
height control wanted
enable height control
[GPSmode Wanted]
switch on GPS mode
GPS mode wanted
enable height control
enable GPS mode
[no further command]
[steering command]
GPS mode has to be activated for waypoint navigation
[using RC]
change a steering angle or throttle
steering command
[using computer]
[waypoint navigation]
activate waypoint navigation
define waypoints
send waypoints
fly following waypoints
[else]
define steering command
steering command
Change flight state
change flight state

Figure 3. Examples of SysML diagrams for use case "manipulating quadrocopter while flying"

Subsequently, the quadrocopter was modelled using the matrix-based concept of the IFM framework (see Figure 4). Modelling was performed using MS Excel. As the IFM framework and SysML share the concept of use case, these and central, inherent process steps are equal in the IFM framework and the earlier modelled SysML diagrams. This circumstance, however, prevents a direct quantitative comparison of the time needed for modelling with both approaches as use cases and central processes were now already known. For modelling variant process flows for one particular use case, alternative sets of associated views were generated for each alternative, using different *process flow views*. Seven alternative sets of views were generated for the use case *manipulating quadrocopter while flying*.

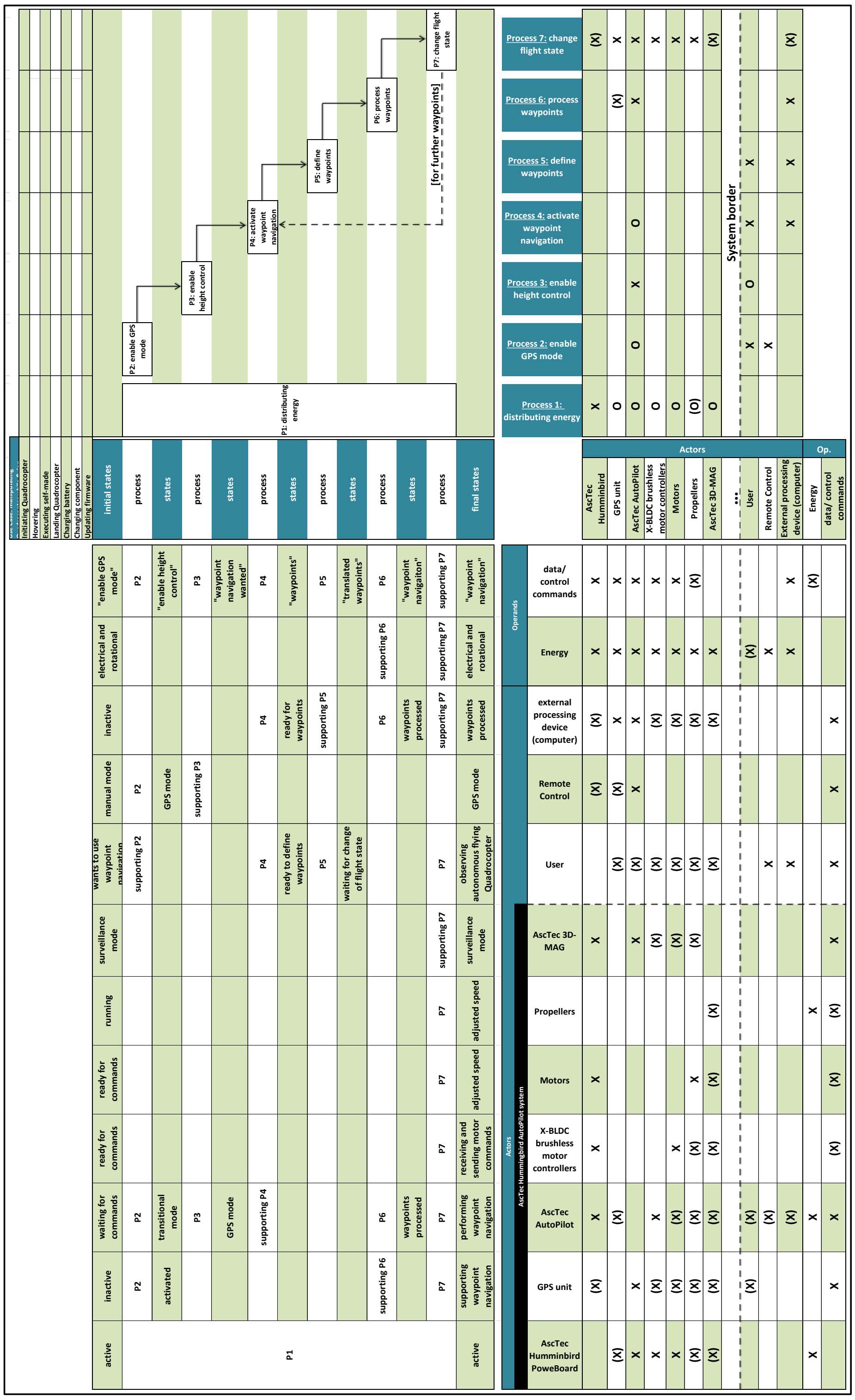


Figure 4. Exemplary sets of views in IFM framework for use case "manipulating quadrocopter while flying" (total amount of sets of views for this use case is seven, based on alternative process flows)

5.2 Main observations

One particular aspect to be compared is the usability of both approaches for function modelling of the quadrocopter. The comparison focuses on the *training efforts*, *modelling efforts* (not including gross modelling time, as discussed before, thus mainly referring to the operations required for generating the final (set of) models), the usability of the inherent *formalism, adaptability* of the approaches, *model readability* and *change management* (i.e. revising the model, e.g. as new information is gained in the process).

- ***Training efforts***: The student mainly responsible for modelling had no prior experience with SysML or the IFM framework. A considerable amount of time was required to learn and get experienced with the SysML vocabulary and formalism before he could get started with modelling the various diagrams in SysML.
- In contrast, the time required to get started with modelling in the IFM framework was considerably shorter as no particular formalism had to be learned and as matrices are a well-known means for representing information allowing modelling to start rather facilely.
- ***Modelling efforts***: One aspect that continuously required attention while modelling in SysML was the implementation and refinement of all formal definitions for the modelled entities, which was not the case in the IFM framework. Another difference concerns modelling alternative process flows (i.e. the variants mentioned earlier). In SysML, *activity* or *sequence diagrams* allow modelling alternative process flows, error scenarios, etc. occurring within a use case into the same diagram (see Figure 3).
- In the IFM framework, each set of views is associated to one particular process flow. Therefore, each variant in a process flow needs to be modelled in a separate set of views. Although copying and adapting already finalised spreadsheets in Excel was quite helpful, modelling efforts were still extensive for use cases that included a large amount of alternative processes.
- ***Formalism***: Creating a formally correct SysML model requires equipping a large amount of model elements with appropriate formal types. Finding and creating the correct/suitable types took a considerable amount of time and continuous reflection. Particularly the gradual refinement and expansion of created diagrams with new entities was found to cause difficulties in this respect.
- Modelling with the IFM framework does not involve such formalism. Thus, no such problems were experienced.
- ***Adaptability***: Adapting the originally proposed views or diagrams, respectively, was experienced to be relatively easily possible in both approaches. In SysML this can be quickly performed by creating new profiles and stereotypes to adapt and expand diagrams wherever needed.
- In the IFM framework, whenever needed, contents or forms (e.g. adding new sub-sections in rows or columns) of individual views could also be quickly adapted to the needs of a specific situation.
- ***Model readability***: Several SysML diagrams quickly became rather complex and thus difficult (and time consuming) to read even for the modellers (i.e. many interconnections, notes, alternatives, etc. had to be represented). Further, the distribution of information across multiple diagrams (which are not always clearly linked) created problems for any person not initially involved in creating the respective diagrams. As can be expected, in these cases, the before-mentioned efforts for adapting diagrams were also considerably larger for SysML compared to the IFM framework.
- In comparison, a set of views in the IFM framework always refers to one specific use case, thus representing all information describing this use case in one spreadsheet. However, maintaining a holistic view of the entire system when modelling multiple use cases is hampered because information is distributed over different spreadsheets. Also, modelling highly complex use cases can result in rather large matrices quickly, which can be difficult to read on a computer screen.
- ***Change management***: A general problem in modelling complex functionality in different views or diagrams, respectively, is ensuring model consistency, i.e. orchestrating any introduced adaptations to associated views/diagrams that will also be affected by a particular change. In SysML this was found to be more difficult than expected. In some cases, the software tool would indeed display warnings if changes in one diagram also affected others, which was very helpful. However, in most cases (particularly if actors and their interactions with others were adapted), no

such warnings were provided, thus, frequently yielding inconsistencies across diagrams. These often went unnoticed until rather late in the modelling process.

In comparison, within a single set of views (i.e. for one particular use case) in the IFM framework, adaptations made to one view could be rather easily orchestrated to all affected adjacent views. However, if the changes affected views in other use cases as well, additional efforts were required to verify their consistency and implement required adaptations. Nevertheless, overall, experienced efforts for implementing adaptations were considerably lower in the IFM framework.

5.3 Discussion

The insights from the literature review and the application of both approaches provided valuable insights. In the following, these are critically evaluated and discussed.

5.3.1 SysML formalism may both hamper and support the modelling process

The strong formalism in SysML was a barrier in the beginning and required continuous efforts throughout the entire modelling process. The initial learning efforts and required abstraction in SysML has already been identified as one of the main barriers for its wide-spread use in interdisciplinary design by other researchers (see e.g. Borches and Bonnema 2010). At the same time, the authors of this paper found that the formalism provides a certain guidance: the formalism predetermines how the different entities should be used, which occasionally was rather helpful in setting up the diagrams.

The IFM framework provides much more freedom, which was particularly beneficial for setting up early models on paper, as these could be quickly adapted and refined. The formalism in SysML prevents such quick adaptation. Eventually, both freedom and formalism thus have their benefits and shortcomings. Whether this is perceived as one or the other depends on personal preferences of a designer and on the context of the project. It may be beneficial to balance properly between formalism and flexibility by adapting both approaches adequately.

5.3.2 Visual and formal linkage of information are beneficial in modelling

One of the strengths of the IFM framework is the direct visual linkage between matrices, which proved particularly beneficial during adaptation of information within and across different views. A remaining problem is the linkage of different sets of views for different use cases. This problem is avoided to a certain extent in SysML through the use of the meta-model. Although some difficulties occurred with this functionality during change management, it is in principle a useful feature. Here, a large potential for cross-fertilisation exists. While SysML could certainly benefit from a clearer visual connection of information across diagrams, the IFM framework could benefit from (somehow formally) linking information across different sets of views. The latter suggests the need to develop an adequate software tool to support such a feature.

5.3.3 Readability of views and diagrams needs improvement

For large SysML diagrams, it proved difficult to arrange all model elements adequately, especially if many connection lines had to be drawn. In this respect, the matrix-based representation in the IFM framework was perceived much more convenient during modelling. However, in case many different entities are to be represented in the IFM framework, the matrices can become very large, making them difficult to read as well. In these cases, also the direct visual linkage between the different matrices (one of the main benefits of the IFM framework) is hampered. This suggests that in order to maintain readability of the different views, it may be beneficial if specific contents could be flexibly highlighted or blanked out in the views, respectively to allow focussing on specific parts only. This, again, suggests implementation of the IFM framework into an adequate software tool.

5.3.4 Representing alternative process flows in the IFM framework needs improvement

The possibilities to represent alternative process flows and error scenarios in one *activity diagram* proved to be beneficial in SysML. Even though the respective diagrams may subsequently become rather complex, these possibilities decrease modelling efforts considerably in comparison to the IFM framework. Variant sets of views in the IFM framework to represent all alternative process flows sometimes only differed by very few process elements. The result is a large number of sets of views required for comprehensive modelling. This, eventually, increases modelling efforts hampers adapting the views quickly if new information is gained.

5.4 Implications

The main insights from the presented comparison of the IFM framework and SysML are the following:

- the IFM framework and SysML both should provide more support to ease readability of complex views or diagrams, respectively;
- in SysML, visual linkage of information across diagrams should be enhanced, while the IFM framework requires support in linking information across different sets of views;
- the IFM framework could benefit from adequate conceptual adaption to allow representing alternative process flows, error scenarios, etc. for a use case in the same set of views.

As discussed before, in order to improve the applicability of the IFM framework, the first and second issue may be addressed by developing an adequate software support. Development of such a software support is part of on-going research. In particular, the issue of how to adequately compromise between the formalism that is required in such tools and maintaining flexibility needs to be researched properly. The most pressing issue, however, seems to be the insufficient support for modelling alternative process flows for use cases. A solution to this issue is presented in the following section.

6 CONCEPTUAL ADAPTATION OF THE IFM FRAMEWORK

The idea behind the developed adaptation of the IFM framework was to integrate the possibilities offered by *activity diagrams* in SysML for representing alternative process flows and error scenarios into the IFM *process flow view*. The visual linkage between the represented elements across adjacently placed views was to be maintained. Hence, in particular the horizontal spread of process blocks in the *process flow view* was to be maintained in order to keep the direct linkage with *use case* and *actor views* (see Figure 2). A particular difficulty arose from representing alternative successive state changes in time in relation to alternative processes in the *state view*. The developed concept is presented in the following. Only the *process flow view* and *state view* required adaptation.
The *process flow view* was expanded with common symbols used in a SysML *activity diagram* (see Legend in Figure 5). Also the notation of so-called "guards" from SysML (i.e. textual statements in square brackets, representing conditions for a particular path to be taken) is integrated. With the help of these elements, alternative processes may easily be modelled next to each other (i.e. still spread horizontally), while allowing their clear visual separation using the different "decision nodes". Alternative processes are equipped with the same process number but have different annotations (e.g. P4, P4' and P4''or P4 and P4*, etc.). Simple termination scenarios (like e.g. "no further command" in Figure 5) are modelled as outgoing flows of a decision node and end in a "final node".

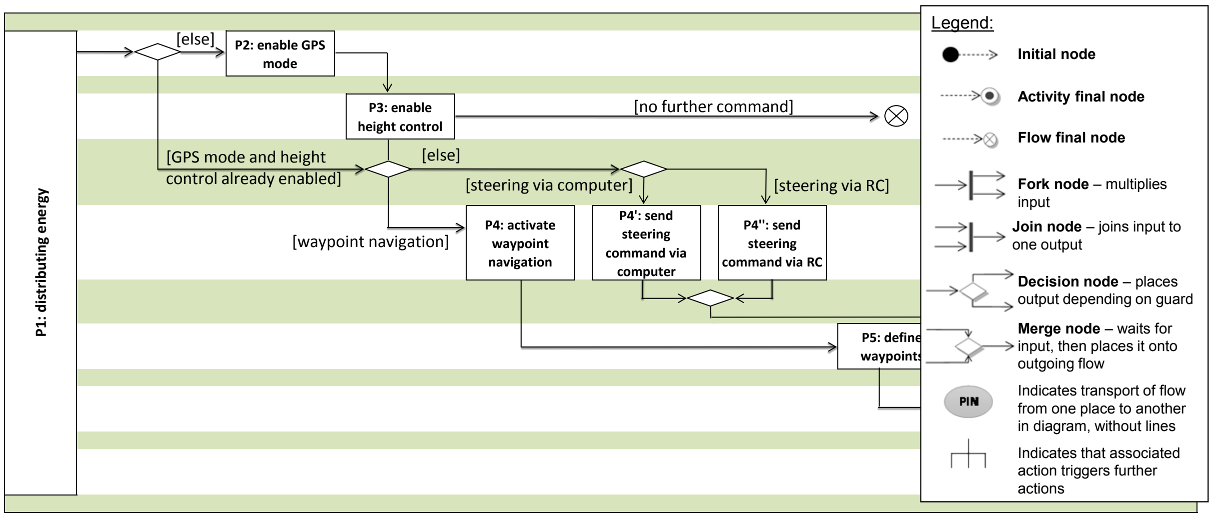
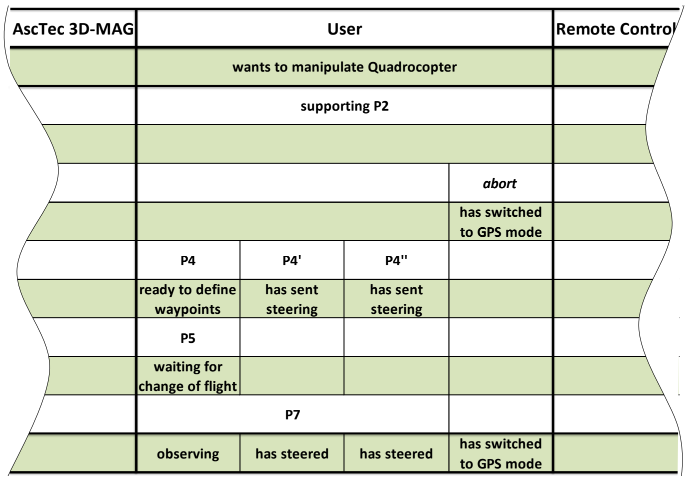

P1: distributing energy
[else]
P2: enable GPS mode
P3: enable height control
[no further command]
[GPS mode and height control already enabled]
[else]
[steering via computer]
[steering via RC]
[waypoint navigation]
P4: activate waypoint navigation
P4': send steering command via computer
P4'': send steering command via RC
P5: define waypoints
Legend:
Initial node
Activity final node
Flow final node
Fork node – multiplies input
Join node – joins input to one output
Decision node – places output depending on guard
Merge node – waits for input, then places it onto outgoing flow
PIN
Indicates transport of flow from one place to another in diagram, without lines
Indicates that associated action triggers further actions

Figure 5. Adapted process flow view containing all alternative process flows for use case "manipulating quadrocopter while flying"

As can be expected, alternative processes may evoke alternative states (and state changes) in operands and/or actors. If such alternative states and sequential changes occur, the respective column in the *state view* is split into different sub-columns. Their representation parallel to the *process flow view* is illustrated in Figure 6 exemplarily focusing on the user. In the given example, the processes P4, P4' and P4'' (see Figure 5) evoke three alternative sequences of states and their changes. These are represented as three alternative sub-columns in the *state view*. Later in the process flow, the alternative paths reunite triggering process P7 (see Figure 5). Therefore, the three corresponding sub-columns (for P4, P4' and P4'') reunite as well (see Figure 6). The fourth sub-column corresponds to the abort scenario after P3 (see Figure 5), which can be similarly modelled using a parallel column (see Figure 6).



AscTec 3D-MAG	User				Remote Control
	wants to manipulate Quadrocopter				
	supporting P2				
				abort	
				has switched to GPS mode	
	P4	P4'	P4''		
	ready to define waypoints	has sent steering	has sent steering		
	P5				
	waiting for change of flight				
	P7				
	observing	has steered	has steered	has switched to GPS mode	

Figure 6. Excerpt of state view indicating state changes corresponding to alternative process flows

The adaptation of the framework was successfully applied while remodelling the quadrocopter and led to a considerable reduction of the number of sets of views. The seven different sets of views required to represent the use case "manipulating quadrocopter while flying" before were effectively reduced to one single set of views. This directly results in a substantial reduction of modelling efforts and of the before-mentioned difficulties with linking information across different sets of views; thus, it also supports building a holistic view of a system's functionality.

7 CONCLUSION

The central aim of the presented research has been the comparison of the IFM framework with SysML in order to derive possibilities for cross-fertilisation and improvement of the IFM framework. Based on both the literature review as well as the obtained insights from modelling the functionality of a quadrocopter, it was derived that both SysML and the IFM framework have their specific advantages and disadvantages. The main differences seem to originate from the strong formalism used in SysML, in contrast to the emphasis on flexibility in the IFM framework. Three main potentials for improving the IFM framework were derived. While two of them are currently being addressed in the development of an adequate software tool implementing the framework, the third, i.e. the issue of representing alternative process flows, required conceptual adaptation. The adaptation of the framework proposed in this paper (i.e. the described expansion of the *process flow view* and the *state view*) solves the identified shortcoming. It was successfully applied for modelling the example of the quadrocopter. With the proposed concept it was possible to model alternative states provoked by the execution of alternative processes, even for very complex process flows. The possibility to do so, in this particular example, resulted in a considerable reduction of the required modelling efforts. It is expected to provide designers with similar benefit in modelling other multi-technology systems. This will be tested in future research. Such research will involve applying the adapted IFM framework for modelling different kinds of complex technical systems in engineering practice to evaluate and improve its applicability further.

ACKNOWLEDGEMENT

The authors would like to thank the *FNR Luxembourg* and the *Australian Research Council* (project DP130101065) for supporting the presented research and the work on this article. Further thanks go to Prof Holger Voos and his research group at the University of Luxembourg for valuable discussions.

REFERENCES

AscTec (2014) User's Manual to Asctec Hummingbird AutoPilot, available online at: http://www.asctec.de/downloads/manuals/AscTec-Autopilot-Manual-v10.pdf (12 April 2014).

Bone, M. and Cloutier, R. (2010) The Current State of Model Based Systems Engineering. Results from the OMG SysML Request for Information 2009, Proceedings of the 8th Conference on Systems Engineering Research.

Borches, P. and Bonnema, G.M. (2010) System Evolution Barriers and How to Overcome Them!, Proceedings of the 8th Conference on Systems Engineering Research.

Chakrabarti, A. and Bligh, T. P. (2001) A Scheme for Functional Reasoning in Conceptual Design, Design Studies Vol. 22, No. 6, pp.493–517.

Dohr, F., Eisenbart, B., Huwig, C., Blessing, L. and Vielhaber, M. (2014) Software Support for the Consistent Transition from Requirements to Functional Modeling to System Simulation, Proceedings of the 10th NordDesign Conference.

Eder, W. and Hosnedl, S. (2008) Design Engineering: A Manual for Enhanced Creativity, CRC Press, Boca Raton, London, New York.

Eisenbart, B. (2014) Supporting Interdisciplinary System Development Through Integrated Function Modelling, Dissertation, University of Luxembourg, Luxembourg.

Eisenbart, B., Blessing, L.T.M. and Gericke, K. (2012) Functional Modelling Perspectives Across Disciplines. A Literature Review, Proceedings of 12th International Design Conference – DESIGN.

Eisenbart, B., Gericke, K. and Blessing, L.T.M. (2013a) An Analysis of Functional Modell Approaches Across Disciplines, AI EDAM, Vol. 27 No. 3, pp. 281–289.

Eisenbart, B., Gericke, K. and Blessing, L. (2013b) Adapting the IFM Framework to Functional Approaches Across Disciplines, Proceedings of the 19th International Conference on Engineering Design – ICED.

Eisenbart, B., Qureshi, A.J., Gericke, K. and Blessing, L.T.M. (2013c) Integrating Different Functional Modelling Perspectives, in Chakrabarti, A. and Prakash, R. (Eds.), Global Product Development, ICoRD'13, Springer, London, pp. 85–97.

Eisenbart, B., Gericke, K. and Blessing, L. (2014) Application of the IFM Framework for Modelling and Analysing System Functionality, Proceedings of the 13th International Design Conference – DESIGN.

Erden, M., Komoto, H., van Beek, T. J., D'Amelio, V., Echavarria, E. and Tomiyama, T. (2008) A Review of Function Modelling: Approaches and Applications, AI EDAM, Vol. 22, pp.147–169.

Frankenberger, E., Birkhofer, H. and Badke-Schaub, P. (Eds.) (1998) "Designers: The Key to Successful Product Development", Springer-Verlag, London.

Friedenthal, S., Moore, A. and Steiner, R. (2006) OMG Systems Modeling Language (OMG SysMLTM) Tutorial, INCOSE - International Symposium.

Kreimeyer, M. and Lindemann, U. (2011) Complexity Metrics in Engineering Design. Managing the Structure of Design Processes, Springer, Berlin.

Kleinsmann, M. and Valkenburg, R.C. (2008) Barriers and Enablers for Creating Shared Understanding in Co-design Projects, Design Studies, Vol. 29 No. 4, pp. 369–386.

OMG (2012) OMG Systems Modeling Language (OMG SysMLTM) Specification, available online at: http://www.omg.org/spec/SysML/1.3 (12 April 2014).

Pahl, G., Beitz, W.F.J. and Grote, K.-H. (2007) Engineering Design: A Systematic Approach, Springer, Berlin, Heidelberg, New York, Tokyo.

Peak, R., Paredis, C., McGinnis, L., Friedenthal, S., Burkhart, R. (2009) Integrating System Design with Simulation and Analysis Using SysML, INCOSE Insight, Vol.12, No.4, pp. 40-44.

Pires, A., Duprat, S., Faure, T., Besseyre, C., Beringuier, J. and Rolland, J. (2012) Use of Modelling Methods and Tools in an Industrial Embedded System Project. Works and Feedback, Proceedings of Embedded Real Time Software and Systems – ERTS.

Shai, O. and Reich, Y. (2004) Infused Design. I. Theory, Research in Engineering Design, Vol. 15 No. 2, pp. 93–107.

Weilkiens, T. (2008) Systems Engineering mit SysML: Modellierung, Analyse, Design, dpunkt.verlag, Heidelberg.

(www1) http://example.system-modeling.com/ (15 March 2014).