A Design Framework for Model-based Development of Complex Systems

Hristina Moneva, Roelof Hamberg, Teade Punter
Embedded Systems Institute (Eindhoven, The Netherlands), e-mail: [firstname.lastname]@esi.nl

***Abstract*—A Design Framework is presented that aims at capturing the design rationales in the process of designing embedded or cyber-physical systems. Its principal concepts cover storing the design rationales, which encompasses design decisions and analysis results, by linking design goals to concrete questions and analysis results for a particular scope of the system. The Design Framework does also provide a mechanism for using heterogeneous models for different system parts and linking them by means of essential design parameters and their dependencies. An elaborated conflict detection mechanism at different levels is provided in order to enable the designer to keep the design consistent throughout the process. The paper also presents first experiences in applying the prototype in industrial contexts.**

***Keywords:** model-based engineering; modeling formalisms; multidisciplinary design; design framework;*

I. Introduction

In the current practice of designing Cyber-Physical Systems (CPS), like MRI scanners and copiers, designers face the challenge of balancing the abilities to create a variety of models for analyzing system options and to track their role in the decision process. This often causes the following questions to pop-up in design processes:

- Why was a model made? Are the underlying assumptions still valid?
- Do the results still hold? Is the version of this model up-to-date?
- How do analysis results relate to design parameters?
- How is the system affected by a design parameter change?
- And many more questions…

The problem of capturing design rationales was already addressed in [1], [2]. We think that the problem is related to the challenges for CPS as observed in [3], i.e. ‘Low productivity of CPS software engineers’. In this paper we present a so-called Design Framework that aims at capturing the design rationales in complex system design. The framework is developed by using our institute’s knowledge about designing complex systems at mainly Dutch-based companies like Océ-Technologies, Philips Healthcare, and Vanderlande Industries.

CPSs do always require the involvement of specialists from multiple disciplines. The multidisciplinary character of complex systems does require different modeling approaches too. Figure 1 shows the multidisciplinary character of a generic design process. For design, a system overview is needed, which is one of the main concerns of system architecting. An architect compares the design options and takes decisions to choose from these which will impact the subsystem disciplines. The system overview directs the variety of disciplines, like the software, mechanical, and electronic hardware. Each of these disciplines, sometimes organized as departments, are bothered with delivering parts of the complete system design. System architecting is also fed by the disciplines themselves, e.g., when new design information within a discipline becomes available. Therefore a Design Framework that captures design rationales should capture overall system overviews as well as overviews within the disciplines to come up with adequate design information and their interactions.

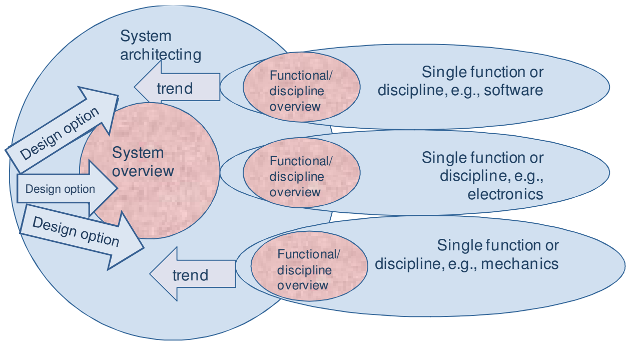

System architecting
Design option
Design option
Design option
System overview
trend
trend
Functional/ discipline overview
Functional/ discipline overview
Functional/ discipline overview
Single function or discipline, e.g., software
Single function or discipline, e.g., electronics
Single function or discipline, e.g., mechanics

Figure 1 Multidisciplinarity in CPS design.

The models that are derived at the overall system level as well as in each of the disciplines are stored and should be related to each other. This is why the design rationale feature of the Design Framework is also denoted as *model and result management*. For relatively stable functions of the system the partial models are increasingly playing the role of starting point for implementation. The specification languages used for this are tailored towards the specific domain: they are called domain-specific languages.

A second motivation to develop the Design Framework is that today’s complex system development requires *modeling* for analysis, see e.g., [4]. The main reason for modeling is rooted in the need to have design questions about a system answered, e.g., a performance model is made to answer a performance issue. We observe that multiple formalisms (the languages to make the models) are applied in industrial practice. The need for multiple modeling formalisms stems from their ability to answer different design questions. We also observe that CPSs are hardly modeled in a complete way [1]. Rather only the critical parts are modeled, because it is often not cost effective to model the complete product.

A third motivation for defining the Design Framework is the presence of *concurrent engineering* in design processes. All specialists from diverse teams are designing the same system simultaneously and the communication between them is crucial for early error detection, which is usually hampered by implicit assumptions and failing communication of design decisions. Many organizations have organized their processes according to well-structured process approaches like RUP[1], DO-178B[2] and even may apply the CMMi[3] or ISO/IEC 15504 standards to assess their process maturity. On the other hand, we have observed organizations that develop complex systems, but do not have an explicit process in place at the detailed design level; in such cases only the main milestones are well defined. The Design Framework aims at supporting different implementations of processes. It therefore has a very simple and basic concept of a process step, i.e. the generic design step (see Section II).

The remainder of this paper first introduces the concepts of the Design Framework and its prototype tool in Section II. Next, experiences while applying the Design Framework are shown in Section III. Finally, an outlook for further development on the Design Framework is given and the paper is concluded in Sections IV and V, respectively.

II. Design Framework

In this section, we present the reasoning that led to the proposed approach, viz. a model that is targeting the support of a model-based design process as we believe it should be, i.e., with partial modeling in order to address the relevant design issues, as well as showing how it relates to current industrial practice. In the end, a prototype that embraces these concepts is presented.

A. *The model*

One way of classifying the variety of activities in a design process is according to their level of formality. On the one hand, more formal activities exist, which cover aspects like creating, manipulating, and analyzing concrete models. On the other hand, the more informal activities describe all the development steps that are made and the design decisions that are taken. These steps and decisions put all models in the context of the desired system. The informal activities can also be further distinguished: they are either activities related to the concrete design process or activities related to the design itself.

In order to support all design activities in a design process, a generic Design Framework model is proposed, see Figure 2. The Design Framework consists of three abstract levels or layers: design flow, design views, and models.

Each level incorporates the process and status aspects. The framework is designed to be generic and to be able to fit in any existent design process or development life cycle, any discipline or specific view, and to enable the usage of any formalism required in the working process. Due to the generality of the approach, multiple features can be added to support the designer solely based on the data, which is already present in the model – conflict detection, impact of a design decision, design exploration, etc. The ability to tailor this framework to the needs of a specific environment is considered an important usability issue. Some of these features will be introduced in Section IV.

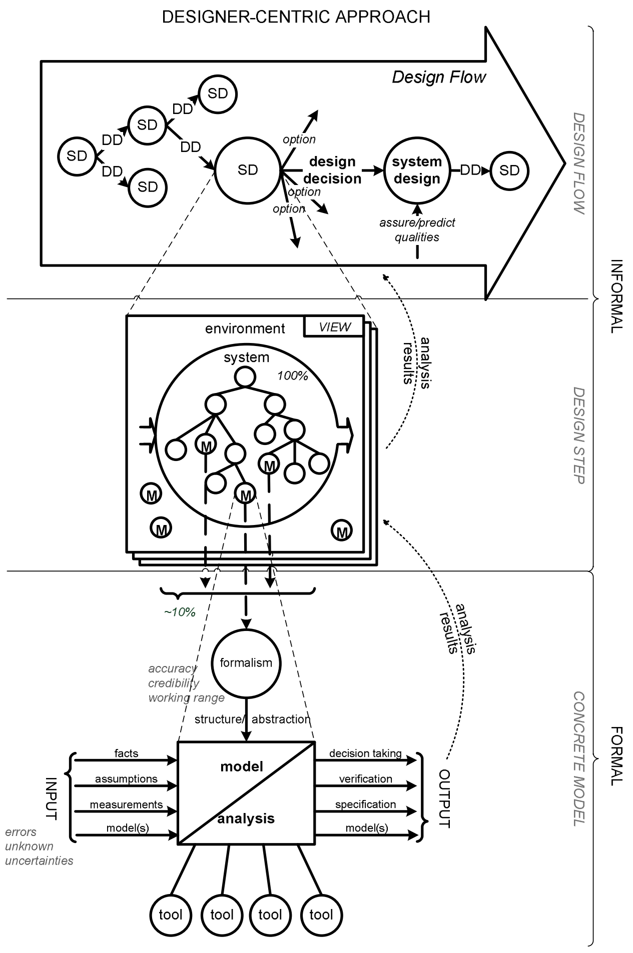


Figure 2. Design Framework model

1) *The top layer: Design flow*

Designing as an activity can be described by two main ingredients – taking a design decision, and, based on it, refining the designed system until the moment in time that a set of options enforces a next design decision to be taken (see Figure 3).

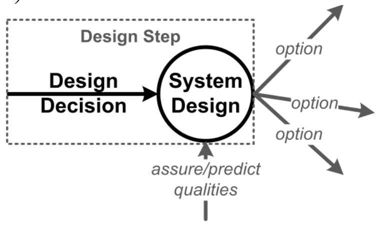


Figure 3 Design decision and design step

If we try to represent this process, it will be tree-like and comprise of a number of nodes (system designs) and

1 http://www-01.ibm.com/software/awdtools/rup/
2 http://www.esterel-technologies.com/do-178b/
3 http://www.sei.cmu.edu/cmmi/

edges (design decisions or options) – see Figure 2, design flow level. Often, some design activities do not lead to the desired system (either dead branches/dead ends, or some possibilities are not explored further (yet)), alternatives, or the designers work on a few options in parallel in order to gain better in-depth insight in particular aspects of each option.

2) The middle layer: Design views

A design view may be considered as a specific representation of the system. It might represent a discipline, e.g., software, hardware, mechatronics, electronics, and material flow, or a specific aspect, e.g., performance, safety. Every view bears a unique decomposition of the system under design – see Figure 2, design view level. Structural basic blocks are used for the decomposition. They form the skeleton of the view. These blocks may contain or may be contained in other blocks.

The basic blocks are used as a container of the detailed models (see Figure 4). The blocks are formalism-independent and consider the model from a black box perspective. One block may contain multiple models, each of which is developed in order to analyze a specific concern or quality of the system or parts of it. If multiple models are present in one block, they are not considered as redundant, because they may model different aspects of the same system block. The necessity of being able to accommodate multiple models is also based on the need of answering multiple design questions by complementary analyses and the inability of any specific formalism to cover all required aspects. Note that multiple system aspects can be dealt with in one view (with multiple models) or in multiple views (each with its own model). It is up to the design team to decide which representation best fits their needs.

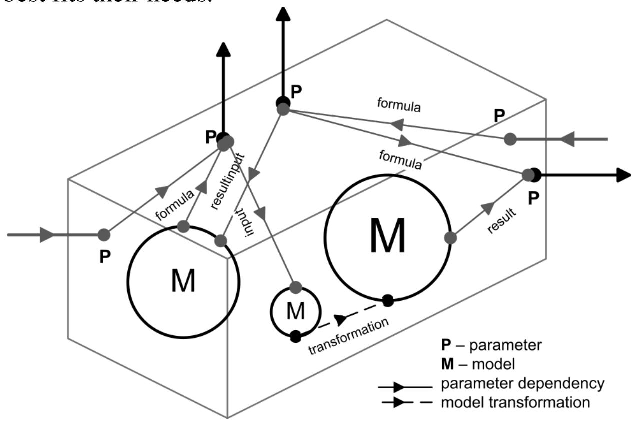


Figure 4 System block

Every block is characterized with a set of parameters. Each parameter might have a (range of) value(s) and a unit, and may have dependencies to other parameters. The parameters are used as a mechanism to couple blocks either within one or between multiple views. The interoperability of multiple formalisms and models is also enabled through these parameters. The basic block and the parameters that characterize it are used as an abstraction of the real detailed models. Due to the parameter dependencies, conflict detection between concrete parameters and their values – inputs or outputs of various models and their analysis results – becomes possible.

3) The base layer: Models

The need of modeling arises from the need of gaining in-depth knowledge of the system under design or its parts. Each model is expressed in its own formalism, which is suitable for analysis of specific aspects/questions. The model – see Figure 2, model level – usually has a number of inputs: a set of facts, assumptions, measurements, or even other models due to model transformations. The model, dependent on a chosen formalism and its degree of abstraction, having inputs with their errors, unknowns, and uncertainties, inevitably also has its own accuracy, credibility, and working range [5]. These are relevant attributes that should be known to the designer.

In general, multiple experiments can and will be performed on one model. The type of experiment is limited by the set of tools and their abilities. The decision to store the results of a particular experiment is under control of the designer. With any experiment, data should be stored about the tool that is used, its parameters, and the results. The results may be used for further decision taking, verification of assumptions or system qualities, or specification of some system parts. A specific result may be another model which in its turn is used for further analysis in case of model transformations. The latter allows for relating the design framework to so-called tool-chains.

4) Horizontal decomposition of the model

The vertical layering of the model can be naturally extended by a horizontal decomposition, see Figure 5. This decomposition is preserving the dual nature of design in itself: design process/activity versus design result/status, or in other words the 'why' and the 'what' aspects of the system under design. In the realm of this reasoning each horizontal layer can be interpreted as 'how' the activity can be performed or 'how' the design status can be detailed even more.

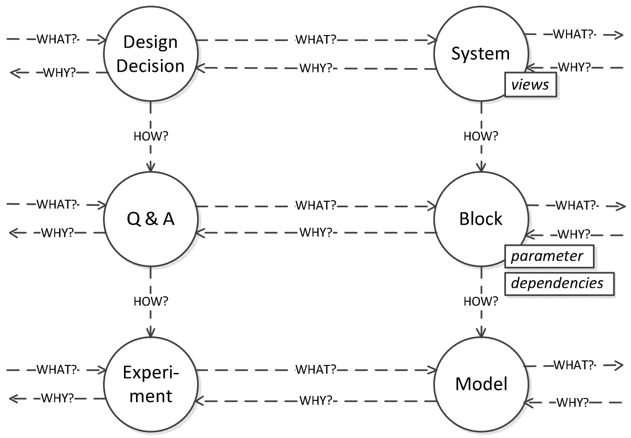


Figure 5 Design activities –'why & what & how' design reasoning

A typical design process involves taking design decisions, detailing the design, and checking whether or not the designed system meets all requirements. These decisions result in a design status of the system design at this point in time (the ‘what’), while the reason for the design status is the design decision itself (the ‘why’). In order to understand the implication of a design decision, relevant questions should be posed and their answers evaluated (the ‘how’). Usually such questions have a particular scope within the system – a particular system block. The relation between the system design and the system blocks is described by ‘how’ they are related to each other (the concrete system decomposition).

Similarly, system block characteristics are the reason for posing particular questions and vice versa. Often, in order to answer a design question, further modeling is required. The model can be analyzed in an experiment and its results will be supportive to providing answers. Hence, the experiment is seen as detailing the design question and the model as detailing the system block.

At all levels top-down and bottom-up reasoning is combined in reality. For example, the need for a particular experiment might predetermine the type of model to be built, but alternatively the model might constrain the type of experiment that can be done with it.

The explained design reasoning concerns just a single design step in the text above. In practice, multiple steps are required for the entire design process of a system: so many times this design reasoning will be applied.

5) *Core domain knowledge*

Most of the system developments are not a greenfield. That is, a lot of knowledge from other projects, from the system designers themselves, and common knowledge is already available from the start. Moreover, a known set of modeling tools is at hand for the developers to support them in their modeling activities. The presented design framework offers a means to store this information as well.

The main reason for including this in the framework is to be able to manage the interactions with changes in the core domain knowledge such as evolving knowledge resulting in adapted system patterns, and different versions of tools. This is a suitable way to store information about reusable library components as well.

B. *The Design Framework prototype*

These concepts were used as a starting point for developing a tool prototype. The prototype is based on a domain-specific language (DSL), also called meta-model, that represents the structure and relationships of all concept elements. This DSL was developed with the help of the Eclipse Modeling Framework[4], it being used for code generation in order to support rapid prototyping and early, continuous testing. Based on this meta-model also two graphical editors were developed, partially automatically generated by the Graphical Modeling Project[5].

The first editor is supporting the user when taking design decisions and storing the design rationales in the form of design questions and answers which are linked to the analysis results that support the conclusions. Moreover, the editor shows the current system design in term of views. The consecutive design steps result in a directed graph, where each node represents a design step. This editor is called the FLOW editor (see Figure 6). From each of the views, a second editor can be initiated. It is meant to represent the system structure and its parameters, dependencies, models, and experiments that belong to one view. This is the VIEW editor (see Figure 6).

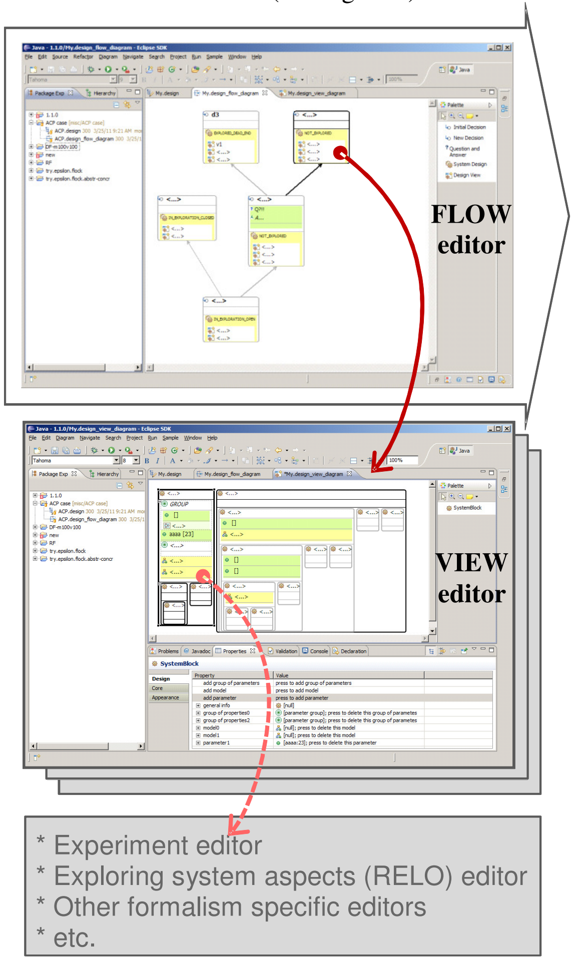

FLOW editor
VIEW editor
* Experiment editor
* Exploring system aspects (RELO) editor
* Other formalism specific editors
* etc.

Figure 6 Design Framework– the basic set of graphical editors

Both editors allow to print the graphical representation which can facilitate various technical meetings by providing the most relevant and up-to-date status of the system under design.

The data of the Design Framework model is stored in a specific XML file, which serves as an input to all graphical editors. That allows easy sharing of a single “design” file among the development team and locally regenerating all graphical representations. It is recommended that along the design itself also all model files will be shared with the entire team.

Currently, the two graphical editors are part of the prototype, but a number of other possible editors are expected to become available (see Figure 6, bottom part). One such editor would support easier manipulation of

[4] http://www.eclipse.org/modeling/emf/
[5] http://www.eclipse.org/modeling/gmp/

model experiments – from more convenient way to store the data to possible automation of the execution. Another editor would support creating and storing exploration traces or diverse system aspects using the relationship-based exploration (RELO) approach, which is discussed in more detail in [6].

III. Experiences in applying the Design Framework

Two industrial studies are presented to provide evidence that real industrial processes can be mapped to the concepts of the Design Framework. Moreover, the studies support one of our claims, i.e., that the design reasoning, one of the foundations of the Design Framework, can be observed in practice.

A. *Architecture team discussions*

In order to test the concepts as well as the prototype, a case study of an actual design process of one of our industrial partners was performed. The development of a new candidate for a customer system was already in place for a long period and still continues at the moment we are writing this paper. One of the authors participated in the weekly architect meetings and has observed the entire process in a period of 2-3 months. This served as an input for the Design Framework prototype and allowed us to gain experience with a real industrial problem at hand.

1) *Case introduction*

The result of the observations was the representation of the design process consisting of five design steps, see Figure 7. The system design – in terms of views, parameters, dependencies, models, and experiments –was getting enriched in each of these steps and was evolving over time. Each design step's goal was detailed in a number of design questions to be answered, which were addressing diverse system aspects and parts.

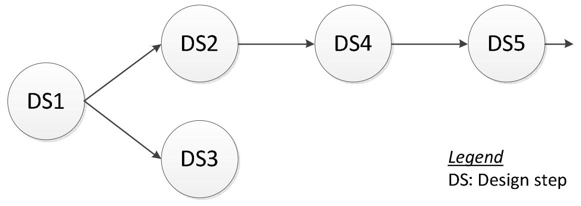


Figure 7 Abstraction of the design steps taken

At the moment when we joined the design process, the architectural team was discussing five possible candidates (DS1). As a consequence, in the first design step we had five separate views which represent these candidates. As these candidates share their basic equipment blocks with identical parameters, we also introduced a separate equipment view. A view with scenarios, which were used as test cases for these candidates, was added as well. These scenarios were based on customer characteristics and allow testing the suitability of the candidates for different market segments. Finally, the evaluation view was added, in which all important system KPIs are specified as well as their estimates per scenario and per candidate in order to compare the available options.

If we zoom in on a view, we see a graphical representation of the skeleton structure and an overview of all important artifacts of each block – parameters, their dependencies, models, and their analyses. For example in the case of candidate C4, the view has a system level block with only one additional level of decomposition into subsystems. Further decomposition at this point is not required, but can be added later. Its depth may vary at different places. At the system level three parameters are used to characterize some of the system aspects, where for example the cost per unit is dependent on the analysis results of two models. Also a number of models and their experiments are stored. The visual overview does only provide names of experiments, while all details – such as the experiment inputs and results, model path, tool, etc. – are accessible in the properties view of each block. The analyses results are used to formulate answers to the design questions of this current design step.

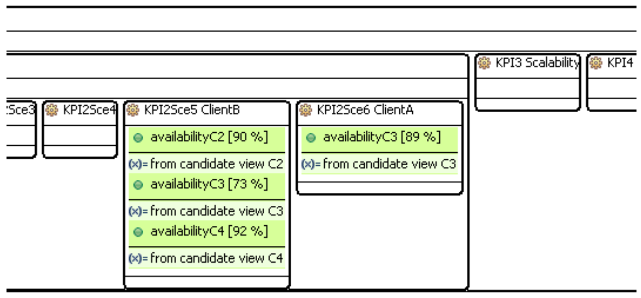
After studying the availability aspect of the candidates and their time-to-market, a number of candidates were discarded. This resulted into a new design step (DS2), where the number of candidates were limited, while other aspects of the remaining candidates were studied. In the meantime, one of the discarded candidates was still being evaluated as a possible solution for a customer by another set of people (DS3). It turned out that this candidate was very suitable to this type of customer and it should therefore be preserved. In order to deal with the different candidates, the architectural team decided for a more generic system, which could be configured by a number of design parameters into each of the three feasible candidate solutions.

This brought us to a next design step (DS4), in which the generic system's performance is examined. Meanwhile another aspect became dominant: a visualization of the systems in this domain was very important and a limited feasibility study was performed. Once the structure of the generic system was fixed, the design activities focused on the control aspects of the system (DS5), where other design concerns play a role. At this point our participation in the design process has ended.

Just to give an impression of the size of a typical design step in this study, the number of models used was 12 and they were referenced 21 times, 16 experiments, 85 system blocks for all views, 139 parameters, and 22 dependencies. The total number of design questions for all five steps was 13.

2) *Observations*

Based on the experience we gained while following the industrial design process, a number of observations are made. The first is that when using the Design Framework, it is easier to track the status of the design activities. For example, in the first design step, the architectural team intended to explore all possible options of candidates and scenarios and study their availability characteristics. For that purpose the evaluation view was created. When you see its status in the VIEW editor, it is obvious that this intention was not carried out fully (see Figure 8).


Sce3 | KPI2Sce4 | KPI2Sce5 ClientB | KPI2Sce6 ClientA | KPI3 Scalability | KPI4

KPI2Sce5 ClientB:
availabilityC2 [90 %]
from candidate view C2
availabilityC3 [73 %]
from candidate view C3
availabilityC4 [92 %]
from candidate view C4

KPI2Sce6 ClientA:
availabilityC3 [89 %]
from candidate view C3

Figure 8 Evaluation view, only partially filled out

A second observation is the fact that some implicit design activities become more explicit when using the Design Framework. A good example is the fact that even though one of the candidates was discontinued, other people went on using this candidate in preparing a customer quotation. Just looking at the top level FLOW diagram, the status of the design and its top-level design decisions become evident, and it is easier to keep everyone up-to-date. In this respect, it is interesting that even less structured processes are fitting the proposed candidates well and can benefit from the proposed methodology to achieve better transparency and more explicit, stronger reasoning.

The third observation is that while following the design process, we observe also a pattern in the design activities, which is strongly related to our underlying design reasoning (see Figure 5). All design discussions were following the proposed pattern, but due to the fact that the architectural meetings were organized on a weekly basis there were a lot of activities that were happening in the background. The role of the meetings was to steer this process and take the main decisions.

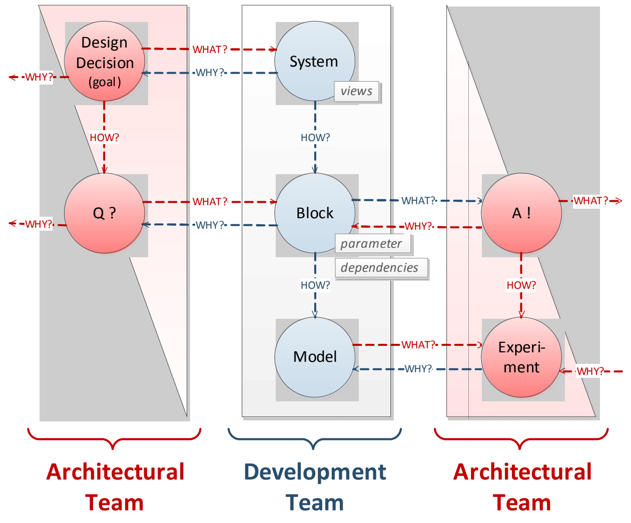


Figure 9 Elementary building block of the design process split according to responsibilities of architects and developers

A decision process starts with the formulation of a goal, based on the current status of the system under design (see Figure 9). This goal can be decomposed into a set of design questions, where each question has its own scope within the system. In order to answer these questions, a set of analyses will be defined, which also indicates the types of models that can be used to achieve these purposes. These activities belong to the architects' responsibilities. The development team is responsible for the actual development of the models and their analyses. These analyses will serve as preparation for the next architectural team meeting, where answer(s) based on the results can be formulated in an evidence-based manner.

Following the scope of the responsibilities of the architects, an architectural meeting has to focus on (see Figure 10):

1. Analyzing the experimental results (based on real measurements and/or models),
2. Answering the design questions from the previous meeting,
3. Identifying the next design goal and its questions, and
4. Defining the required models and experiments in order to prepare for the next meeting.

It can be imagined that the Design Framework tool can be used even during these meetings to structure them as well as to bring all up-to-date results into the room.

Figure 10 The focal steps of an architectural meeting

The fourth observation concerns the fact that the design process had to deal with a number of system candidates and all of those were simultaneously present in the system design process for some time. We choose to represent these candidates as separate views within the same design step. Another approach could have been to represent each candidate by a separate design step, together forming parallel branches in the design process. The approach we selected – representing each option as a view – has the advantage of allowing the candidates to share information available in the equipment view. If the second option had been selected – each candidate with its own design branch – the candidates could only have shared information via the core domain knowledge, which could also have been used to store the information about the equipment. Both approaches are feasible and equally suitable.

B. *Architecture overviews*

A second experience stems from a feasibility study in mapping an architecture overview upon the Design Framework. The architecture overview was defined to capture and to share architectural knowledge about the system latency aspects of one of the machines of one of our industrial partners. The overview was defined with help of the "A3 Architecture Overviews" method [2].

This method aims at capturing the information otherwise available in multiple and scattered documents and models as well as in people's minds. The method results into an A3-sized sheet of paper, where both sides are used – one with more textual and context information, the other with more graphical and model information. The overview has a few key elements, such as functional flow, physical view, system concerns, key parameters and requirements, design strategies, design decisions, assumptions, and known issues. A feasibility study was set up with the goal to follow the creation process of A3 overviews and to store the available design information in the Design Framework.

1) *Case introduction*

Creating A3 overviews is a time-consuming process that spans multiple iterations of creating its contents and communicating its status in order to validate it. Typically, in each iteration different design aspects are being questioned. The reason is that it is not simple how to select the most relevant information from various sources, which have different appraisals due to a range of perceptions and judgements. At the moment of writing this paper the overview creation process was still in progress.

The first version of the architecture overview was focused on the visual representation of the system functional flow and physical view along with a few of the key parameters and system constraints. The mapping to the Design Framework concepts was trivial. The functional flow and the physical view were represented as design views within the Design Framework. The flow decomposition was depicted by a hierarchy of system blocks. Each visual representation of some of the flow steps became a model attached to the corresponding system block. A similar mapping principle was employed for the physical view. The key parameters section contained a table with values, which was represented by parameters belonging to specific system blocks. The system constraints section was also translated into a set of parameters, but some of them also had dependencies to other parameters.

The successive versions of the A3 overviews contained more elaborated design information in terms of models. Some of the visual representations of flow steps were updated and more key parameters and system constraints were introduced. That resulted into a new design step in the Design Framework, where some models, parameters, and dependencies were updated and others were introduced. This version also had a partially filled text side of the overview, where some definitions of system parameters in different views were introduced as well as some system requirements and domain constraints. All these were also mapped to various models, parameters, and dependencies.

2) *Observations*

The main observation is the ease of mapping of the concepts of the A3 method and the Design Framework. While these two approaches have different goals, they are both trying to reveal and represent the design rationale of a system. That strengthens our confidence in the approach presented in this paper.

Another interesting observation is that while mapping the system constraints and design decisions there was design reasoning containing "because" in its phrasing. In order not to lose this design rationale, we mapped it to design questions and provided the reasoning as an answer to it. That allowed us making the design rationale even more explicit than just being hidden among multiple parameter values put in a textual form. Each design decision had its own goal of elaborating or understanding specific parts or aspects of the system in a better way. With the Design Framework it was possible to provide the design rationale in the form of questions and answers along its goal – not the entire set of concerns but just the relevant ones.

Knowing that the Design Framework and the A3 overviews can be easily mapped to each other, we think that the information stored in the Design Framework may be used for generating such architectural overviews. It is important to note that this type of overviews is meant to present just a limited but relevant set of information, while the initial data set may be much larger – coming directly from the architects themselves (the original approach) or from the Design Framework. At this point the challenge would be finding appropriate techniques 1) to select the information that is to be used and 2) to change its visual appearance to the best suited form for automatically created overviews.

IV. OUTLOOK

The two industrial cases for applying the Design Framework prototype show the applicability of the presented concepts and the reasoning framework behind them. All observations in terms of process and produced artifacts were seamlessly mapped onto the concepts.

However, the Design Framework is more than that. There are a number of features that provide even better support to its users [6]. These features can automatically be harvested to support design activities as a result of the generality of the presented concepts and the incremental actual data that is provided by the user in the course of designing. Here we list those features:

- Conflict detection. Based on the data that is being stored in the framework during designing, a continuous conflict detection mechanism can be employed to support the user. The mechanism applies to parameters, their values, and their dependencies and it will continuously check for incompatibilities. That also includes model versions and experiment inputs and results.

- Decision tree and impact of design decisions. An important feature is the ability to deduce the impact of each design decision on the system under design. That can be obtained by differencing any pair of design steps and observing the changes in the designs. This feature facilitates retrospection of the decisions taken and their exact impact on the system. Often, when taking a decision this does not imply that designers are aware of the changes required to the design in advance. In such cases retrospection may help to improve the understanding as well as to give space for improvements when necessary.
- Exploration or cause-effect analysis. This feature, also called relationship-based exploration (RELO), allows expanding interactively a different visualization, starting from any parameter and continuing the exploration in any direction based on the provided dependencies to other parameters. We can use this for showing and storing specific aspects of the system under design in cases where views reflect disciplines and the system qualities/concerns are spread among them (e.g. availability, performance).
- Other features concern different representations of design activities based on time or based on the stability of system parts, reuse of information in different forms, collaborative work and meta-processes, as well as tailoring the concepts and tool to custom domain needs.

The current and future activities target the completion and maturity of the abovementioned features, which will enable us to validate the full potential of the presented approach. While these efforts are still in progress and are based on the confidence we obtained while applying the current version of the Design Framework prototype on various industrial problems, we focus now on real-time application of our tool inside the industry.

Other activities will be focused on investigating the feasibility of this approach for prescriptive processes as well. Such repetitive processes can be found in some domains, such as aerospace and chemical industries, but also in different lifecycle phases, such as sales and certification. Another type of investigation that we envision is on applying the Design Framework in areas as design space exploration and set-based design. Last but not least, we would like to find out more about the different roles of people involved in design – sales, architects, developers, testers, integrators – and their primary interests and possible benefits when using our approach.

V. Conclusions

We have shown the concepts of a Design Framework that supports development in industrial contexts and have presented observations that show its feasibility and relevance. The added value of the Design Framework approach compared to the existing way-of-working of developing CPSs is:

1. Integrated support of design activities, covering three related, but different levels – process (design flow), system decomposition (views), and models and analyses.
2. Characteristic of the approach is its flexibility of using different tools and formalisms, which enables designers and architects to deal with model heterogeneity,
3. Design-time error detection by keeping the dependencies explicit between design decisions, system elements, and models,
4. Support for incomplete modeling, and
5. Ability to show the impact of each design decision on the system under design.

Points three and five are still to be validated in industrial environments.

The framework is domain independent and can be applied in a variety of industries when designing cyber-physical systems in a model-based way. Along all its advantages, there are also some concerns as well. The main difficulty of applying it in industry is the need of introducing a new way of working, which involves exercising more discipline and tidiness and may not be easily accepted. We expect, and are supported in this by our observations, that this will pay off with reduced time for communication, less implicit decisions, and a good overview to all members involved in the design process.

Acknowledgment

This work has been performed as part of the "Integrated Multi-formalism Tool Support for the Design of Networked Embedded Control Systems" (MULTIFORM) project, supported by the Seventh Research Framework Programme of the European Commission.

Grant agreement number: INFSO-ICT-224249.

References

[1] W. Heemels, E. van de Waal, G. Muller, A multi-disciplinary and modelbased design methodology for high-tech systems, in: Conference on Systems Engineering Research (CSER 2006), Los Angeles, CA.

[2] G. Bonnema, P. Borches, Design with overview – how to survive in complex organizations, in: Proceedings of the International Council on Systems Engineering International Symposium (INCOSE 2008).

[3] K.H. Kim, Challenges and Future Directions of Cyber-Physical Software, in: 34th Annual IEEE Computer Software and Applications Conference (COMPSAC), pp.10-13, 2010. DOI: 10.1109/COMPSAC.2010.89.

[4] D. Schmidt, Cover feature – Model Driven Engineering. IEEE Computer, February 2006, pp.25-31.

[5] G. Muller, Coupling enterprise and technology by a compact and specific architecture overview, in: Proceeedings of the International Council on Systems Engineering International Symposium (INCOSE 2007), San Diego.

[6] H. Moneva, R. Hamberg, T. Punter, Are you aware of the design decisions? On how modeling should support design, In: Proceedings of 1st Workshop on Model-based Engineering for Embedded Systems Design (M-Bed 2010) at DATE 2010, Dresden, Germany, 12th of March 2010.