Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 28 (2014) 171 – 178

Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

Early insight in systems design through modeling and simulation

Steven P. Haveman[a,]*, G. Maarten Bonnema[a], Freek G.B. van den Berg[b]

[a]*Laboratory of Design, Production and Management, Faculty of ET, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*
[b]*Design and Analysis of Communication Systems, Faculty of EEMCS, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*

Abstract

In early design stages, system architects mostly rely on estimations to make design decisions. These are based on the available information at hand and their experience. Modeling and simulation is almost exclusively applied in more detailed stages of design. In this paper we present an approach aimed at making better informed design decisions, early in the design process.
Our approach focuses on giving insights in early design through simulations and models that are usually only provided in more detailed design stages. To do so, we propose a framework and address three conflicts that arise when connecting techniques from early and detailed design stages. These are dealing with uncertainty, accommodating multidisciplinary views and accounting for more divergent design space exploration strategies.
The approach has been applied to a medical imaging system, to analyze a possible latency reduction. The goal of this case study was to gain realistic insight in system latency using a highly abstracted system model and a generic simulation model. Insights gained with these models confirmed that a new design reduces system latency and deals better with large variations in latency. The underlying structure of the approach has proven itself to be feasible. Further research is necessary to determine whether the approach can cover a broader range of applications and to evaluate how the full approach can be implemented.

Keywords: Early insight, modelling and simulation, early systems design, latency performance

* Corresponding author. Tel.: +31-53-489-3172; fax: +31-53-489-3631.
E-mail address: s.haveman@utwente.nl

doi:10.1016/j.procs.2014.03.022

1. Introduction

Nowadays, modeling and simulation are common practices in multidisciplinary complex system design and provide system architects with appropriate insights. However, in previous work[1], we identified that early stages of system design lack modeling and simulation, leading to less insight when making early design decisions.

In this work we aim to support system architects by connecting information and techniques that are available at the start of a new design process to simulations and models used in more detailed design stages. Based on a design problem at hand, a system architect will seek insight in certain key characteristics or behaviors of a system. To gain these insights, we propose an approach that draws on the strengths of both early and detailed design. We have applied this modeling approach in a tool that makes it possible to define a high level system model quickly and effectively. For this, we use the Y-Chart paradigm[2] which connects an application model to a platform model through a mapping. The high level model is supported with simulation models that provide the required insight. This approach was applied in a case study on a medical imaging system.

This paper is structured as follows. In the rest of this section, we discuss the background and define our objective. In section 2, we explain the framework and methods in our approach. Then, section 3 outlines a case study in which we implemented our approach. Finally, section 4 contains the discussion and future work.

1.1. Insight in Early System Design

Every design process starts with a collection of knowledge that is both explicit (design documents, requirements documents, an existing system, etc.) and tacit (knowledge and experience of architects and designers). A method that handles both types of knowledge well is the A3 Architecture Overview method (A3AO)[3]. This method has been developed as a tool for effective documentation and communication of architectural knowledge. By combining various views, it creates a concise and usable overview of information of key system aspects on two sides of an A3 paper (or a set of A3's). It has also been shown that this method can provide a good overview of designs during development[4]. However, when the A3 is utilized in design discussions, many designers have expressed a need for more interactivity. This is because a system architect has various ways to reason about a system under design, for instance using thinking tracks[5]. An example of a thinking track is dynamic thinking, which encompasses thinking about aspects such as how the system changes over time and effects of changes in input and output. These thinking tracks give a framework for the insight system architects seek and explain the need for more interactivity, whereas a static system overview gives little insight in its dynamic behavior.

When an architect explores various solutions to a design problem, the architect will make estimations to gain insight in performance or behavior of possible solutions. These estimations are based on experience and are made through logical reasoning, interpolation and extrapolation or through low-order models. Once an architect comes up with a promising design, it will be detailed to verify whether the design is indeed promising. This elaboration can be done using models or simulations. At these, in our view detailed levels, many simulation frameworks exist[6]. A good example is Metropolis[7]. These simulation and modeling techniques give very good insight in dynamic system behavior or insight in how the system deals with feedback.

1.2. Comparing Early and Detailed Design

To increase the insight of system architects in early design stages, we aim to apply these simulation techniques in early design. However, this is not a straightforward process and several conflicts will need to be addressed.

First of all, there is a large difference in the degree of uncertainty between early and detailed design. In both cases, it is necessary to estimate an input to determine performance. But in early design, parts of the system might still be completely unknown. For example, in early car design, the propulsion method might not be decided yet. This complicates insight in for example driving behavior. In detailed design, using the same car example, the early decision might have been to use an electric drivetrain, but the size of the battery packs is yet unknown. In this case, the general behavior of battery packs is already known, so a model can be established much easier.

Secondly, estimations, happening early in the design, are much more multidisciplinary in nature. This means that the input stems from multiple disciplines, but also that the output of the step has to be communicated to multiple

disciplines as well. This happens for example during design reviews, in which key personnel from various divisions is involved. These reviews tend to be complicated and tedious, caused by a lack of insight that stakeholders have in the system and by the many different viewpoints that have to be accounted for.

Finally, we highlight one more distinct difference between estimations in early design and detailed system level simulations. In early stages of design, design space exploration has a much more explorative nature, whereas the more detailed stages of design are more of a problem solving nature. During this divergent design space exploration, there is a large focus on creative thinking, to define and try out new architectures. This detailed, convergent design space exploration aims to find a system design that solves the problem, i.e. meets the requirements. This type of design space is often more constrained as the focus lies more on optimization, also it is supported much better[6].

1.3. Objective

In this work, we present an approach that gives architects and designers the possibility to gain more insight in dynamic system behavior and key system characteristics during early design by utilizing simulations and modeling principles used in detailed design. We do this by creating a more integrated approach between early estimations and detailed system level simulations. To work towards this integrated approach we focus on the main conflicting characteristics (uncertainty, multidisciplinarity and exploration) between these two ways of working.

2. Approach

As was stated in the previous section, our aim is to facilitate more insight in early design stages using simulations. We do this by supporting thinking tracks such as dynamic thinking, which are enabled by more detailed simulations that for example model system behavior. Our approach takes the A3AO method as a starting point, and uses the Y-chart approach[2]. The Y-chart approach is based on connecting an application view to a platform view through a mapping. The resulting system model is supported with evaluation and analysis tools. A functional (application) view and a physical (platform) view are also represented in the A3 method. Building upon these methods, the framework that underlines our approach is shown in Figure 1. It clearly distinguishes between two domains. First, the problem domain, in which the context is defined for the system under design and later on, solutions are verified against this context including requirements. Second, the solution domain, in which solutions are explored and subsequently appropriate models are established to characterize these solutions.

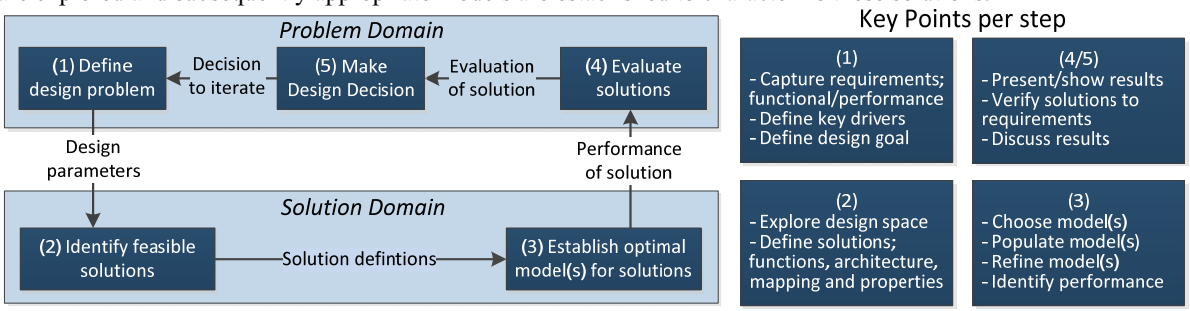

Figure 1. Overview of the approach, adapted from Haveman[1]

In the introduction, three conflicts between early and detailed design were identified. We will now discuss how we deal with these conflicts in our approach.

Dealing with uncertainty; Uncertainty can come in various forms and shapes in early system design. To handle and model uncertainty, both formal and more practical approaches are possible[8]. A common formal approach is to use fuzzy math to account for ranges of estimations. GuideArch[9] is a framework that uses these techniques to guide engineers to make the best choices possible under uncertainty. Another approach is to acknowledge imperfections and uncertainties, and to build your design process to deal with them more effectively. For example, one can use design trees[10] in which feasible solutions can be retraced easily if the chosen solution does not suffice. A more

practical approach, could be to use scenarios[8]. Scenarios can be used to assess possible impacts on a system by envisioning various internal and external influences to the system[11].

However, to our current knowledge, there are few methods that aim to give designers more insight in dynamic system behavior coupled to key system characteristics in early, functional design when multiple system properties are still unknown. The example given in Section 1.2 comes to mind. How can a designer effectively gain insight in the driving behavior of a car if the propulsion method is not yet known? In our approach we combine aspects of the methods mentioned above to address this issue.

Accommodate multidisciplinary views; in the early stages of complex systems design, many stakeholders are involved in the design. Mainly by giving input and posing requirements for the design, but also by giving feedback on new designs. The A3 Architecture Overview method[3] has proven itself as a good multidisciplinary communication tool, by combining various views and using visual aids. Our approach to resolve this main conflict will thus be to utilize the approach and philosophy of the A3 method. Also, we draw on another approach that deals with this conflict by quantifying the impact of design choices on both customer value and other stakeholders value[12]. Here it is observed that a mere technological assessment may overlook the main customer needs. A clear overview of the key drivers of a system is therefore necessary, as they help to focus the development[13]. Architecture-driven quality requirements prioritization[14] is described as a technique to automatically analyze trade-offs between different quality requirements.

Support divergent design space exploration; the final conflict that we want to address in our approach is to account for more explorative design space exploration strategies. However, there is (logically) limited support in detailed modeling techniques to do so. In order to integrate this into our approach, we have to consider approaches that are currently used in early design. An example of a method that supports this kind of design space exploration is FunKey architecting[15]. It applies TRIZ[16] as a structured method for innovative problem solving. An integral part of the FunKey method lies in the fact that it links functions to key drivers and discovers possible architectures through their relation. These functions can be linked to the application view of the Y-chart method.

Another possibility is to use key driver maps to analyze a system[17]. A key driver map links key drivers to requirements via so called application drivers, showing design choices. From this work and from our own experiences[1], we observe that these key driver maps give insight in design reasoning of a system. It shows why requirements exist or are created, allowing designers to question this reasoning and explore alternatives.

3. Implementation

This section focuses on the practical application of our approach. Here, we detail how the system model is represented, establish a structure to interact with lower level models and elaborate on how results are presented, using a case study as an example. This case study concerned the analysis of a redesign of a medical imaging system aiming to reduce latency. The main goal was to evaluate whether it was possible to gain realistic insight in the system latency using a highly abstracted system model and a generic simulation model. The focus on implementing the framework of the approach led to less attention on the identified conflicts. For confidentiality purposes, we have abstracted from specific terms and parameters and describe the case using a generic imaging system.

3.1. Design Problem

Latency is a key aspect in medical imaging and refers to the delay between making an image and the time the physician sees the image on screen. For many interventional procedures, a low latency (~150ms) is required to facilitate proper hand-eye coordination for a physician. In Figure 2 (a), the imaging chain can be seen in the current design. While the current system design meets the requirements, estimations based on the experience of a system architect showed that moving the image realignment functionality from PC1 to PC2 could reduce the total latency and reduce complexity of the system. The estimation of the system architect is based on a summation of estimated latency times of components in the imaging chain. However, insight in the effects of jitter (variation in latency) in the new design could not be established in this way and a need for dynamic insight arose. The case study thus focuses on providing insight for the system architect in the dynamic latency behavior for the two different cases.

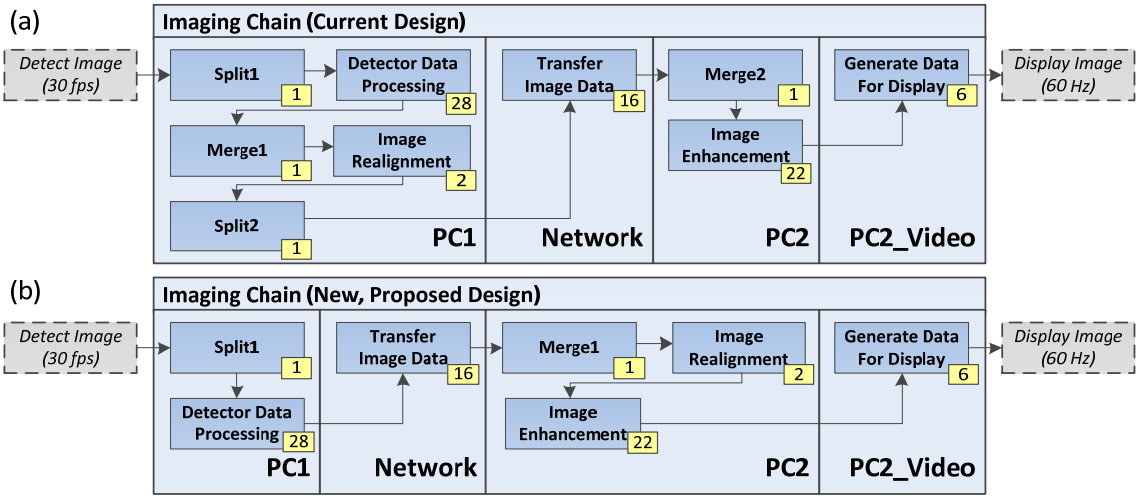

Figure 2. Annotated system models of both designs. The darker and smaller boxes show the functional components, the larger background boxes show the platform components. If a functional component is placed in a certain platform, this means that it is mapped to that particular platform. The numbers show taskload as operations per pixel. All platform components have an operation speed of 1 and are not annotated.

3.2. Identifying feasible solutions

In this case study, the main solutions were outlined from the start. Therefore, the main effort in this part of the case study was put into quantifying the solutions properly. The quantification of components consists of two steps. The first step is to define the units; the second step is to quantify them. In Figure 2, both the designs and the result of this quantification can be seen. Through the Y-chart paradigm[2], we defined the taskload of a functional component as operations per pixel and the operation speed for hardware components as operations per millisecond. Therefore, the division of taskload by the operation speed gives the latency of a component in milliseconds.

The quantification of components can be done using estimations. In our case, detailed latency information was available for several components in the current design situation. However, as it was not possible to decompose this information into taskloads and operation speeds, we quantified the operation speeds of all platform components as 1, and thus assigned the value of the latency fully to the functional component. We are aware that this is a fairly course grained approximation, as we implicitly assume processing capabilities of all hardware to be equal. This approximation is mainly due to the abstract nature of the model content and not due to limitations of the model itself.

3.3. Establishing models

In order to provide more insight in the influence of jitter on the system latency, the right model order needed to be identified, both for the high level system model and the simulation model. A key guideline here is to keep the model as simple as possible, by only including what is necessary. This resulted in an iterative process of refining the models until they were deemed to provide enough insight in the design problem.

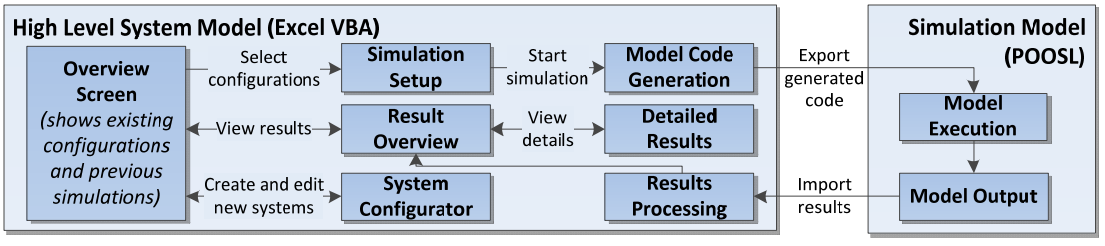

Figure 3. Overview of tool support for the case study

The high level system model constitutes of a list of functional and platform components and their mapping. Several details allow the system definition to be validated. For instance, functions in the high level system model have been detailed with an input and output type, either a full image or only a part of the image. This way, the system can detect whether the input type of a subsequent function differs from the output type of the previous one.

Also for the simulation models, various considerations were made in designing the model. Phenomena such as parallelism or scheduling were not included, because the actual system in our case is configured to handle only one image at the same time in each hardware component. This model was developed using the POOSL language[18][18] and set-up as a generic model to simulate throughput using the Y-chart paradigm.

3.4. Tool support

Both of the models were implemented in a tool (see also Figure 3) to support the system architect in the execution of the approach and to enable automated simulations. The main part of the tool, implemented in Visual Basic for Applications in Excel includes a system configurator and a results viewer. From a single overview, a user can configure new systems, edit existing systems, simulate a system or view results from a previous simulation run. Various simulation parameters can be set up, such as the number of units to run the simulation for and how many runs of the same system should be executed. The tool can also perform analyses using either Excel's built in features or with several custom algorithms.

3.5. Results

As the modeling process was iterative, the modeling results were compared with system architect experiences and expectations often. Each time, both the old system and the proposed system were simulated with multiple simulation runs. An overview of some of these results is shown in Figure 4. The graphs show image latency differentiated per component. Other results were for example latency and jitter statistics. Initially, especially from an outsider's perspective, one would expect the latency to vary. However, when the model was detailed with the display behavior, the results showed a virtually constant latency, as can be seen in Figure 4(a). This is caused by the fact that the variation in latency is 'smoothed' by the time that an image has to wait before the display is ready to show it. In these simulation runs only a handful of images had a different latency, which was exactly one screen refresh rate higher or lower. Further analysis showed that the latency varies between simulation runs depending on the difference between the start times of the generator and the display. In Figure 4(b), the same model as in (a) was used to analyze what would happen if the variation in latency per component increased. This can for example happen if in the future functionalities become more complex and unpredictable, or in the case of resource sharing. The results show a close up of a simulation run and represent the overall results well. These results show that the new design is more stable when exposed to a higher variance in component performance.

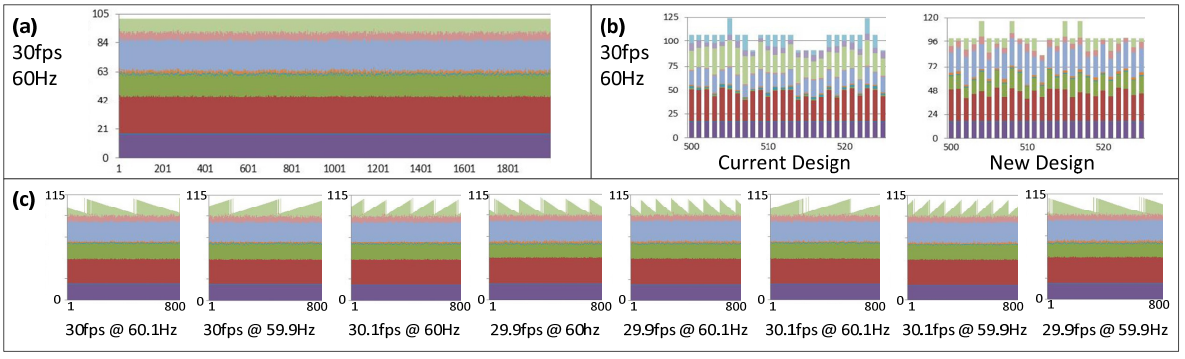

Figure 4. Graphs showing total latency times in ms, composed of latency times for single components (y-axis), for single images (x-axis)
(a) Simulation run of new design with 5% variation in latency per component
(b) Simulation runs with 25% variation in latency per component, zoomed in from image 500 to 525
(c) Simulation runs of new design with varying input rates of the generator and refresh rates of the display

Nevertheless, the results in Figure 4(a) and (b) did not match with the expectations of the system engineers, as actual test results always showed a saw-tooth like behavior. We concluded that the fact that input and output was assumed to be exactly 30fps and 60Hz respectively could be the cause of this. Therefore the high level system model was updated to be able to change the input and output rate slightly. An experimentation with these parameters resulted in various saw-tooth like behaviors as can be seen in Figure 4(c). The exact variance and deviation of input and output currently remains a topic of further research.

Concluding, the results show that the new design does reduce latency and especially performs more stable if the variation is increased. This means that the new design is more future-proof in this aspect. These insights increase the confidence of the system architect in the new design.

4. Discussion

In the previous sections, we have outlined our approach and implemented part of the approach in a tool, which was utilized in a case study. In this section, we will discuss the part of the approach that can be evaluated and extrapolate this evaluation to the complete approach.

4.1. Experiences with implementation and case study

The main goal of creating the implementation and performing the case study was to find out whether dynamic insight in a system can be obtained using a highly abstracted system model in conjunction with a generic simulation model. Using a highly abstracted imaging chain model and a generic simulation model for throughput, we were able to model realistic latency behavior and provide relevant insights. However, a lot of details were necessary to define the models and their behavior. A lot of this knowledge was not documented explicitly as this information is stored "under the hood" in the system model and the simulation model. More transparency and control in this regard are probably necessary, especially in larger or more distributed design teams.

The current case study focused mainly on enabling the approach by creating the underlying framework. Due to the simplicity of the case study, the conflicts outlined in Section 2 received less attention than originally intended. However, we feel that this work is a good foundation to further develop the full approach. A conflict that did return in this case study was dealing with uncertainty. While data could be retrieved to model the current system, amongst others the specific ratio between taskload and operation speed could not be retrieved. Therefore, ranged parameters (fuzzy math) were introduced. These parameters caused a variation in latency that did not match with the expectations of architects, as the latency per component is very stable. This mismatch prompted a modeling decision, as the parameter of a functional component can either be varied or fixed during a simulation run. The latter then requires multiple simulation runs during which parameters are varied. We chose to explore the system while varying the parameter in a single run and investigated specific cases from those simulation runs with fixed parameters.

4.2. Reflection on the approach

In our view, the approach has several key aspects that determine whether or not it will be successful. One of the most important of those aspects is the fact whether an optimal detailed simulation model can be established. An optimal model gives just enough insight for the design problem at hand and is realistic. Determining when and if a model is optimal is the main issue in this aspect. This case study showed that these models actually require a lot of domain specific behavioral information. It also showed that a system architect, through experience, determines whether a model is optimal or not. In our case study, this is reflected by the fact that the simulation results were validated to see whether the behavior was realistic enough.

An issue that is related to this is the fact that making the model actually helps the system architect to understand the design problem at hand. This could mean that once the model is established, the system engineer does not need to for example run simulations to gain the insight, but that the insight is already gained through the modeling process. Based on the single case study that we have performed with this approach, this indeed seems to be the case. However, we feel that the output of the simulations can greatly help multidisciplinary communication, as they are able to make the insight that the system architect gained explicit.

4.3. Future Work

As the described approach is quite extensive, its evaluation cannot be done within a single case study. Therefore, future activities still have to be pursued in various directions. First of all, case studies will be executed that focus on specific conflicts, especially the conflicts that were not represented in this case study. Furthermore, future work will aim towards a good strategy to determine when and if models are optimal. To this end, we propose to use scenario's that help to define the design problem, but also explore whether a model still holds in different contexts. Also we are aware that the current case study was a redesign on a mature architecture. In future work we would like to consider new designs and more novel architectures as well. Finally, in the case study we experienced the logical consequence that a generic model reduces the accuracy, but is more generally applicable and vice versa for more domain specific models. This begs the question whether the models are reusable for similar design problems in the same field, medical imaging systems, or for the same application in other industries, being generic throughput design problems.

Acknowledgements

This publication was supported by the Dutch national program COMMIT, as part of the Allegio Project. To the Embedded Systems Innovation by TNO, as lead partner and to other partners, we would like to express our thanks for their support in making this research possible. Finally, we are grateful for the review comments we received.

References

1. S. P. Haveman, and G. M. Bonnema, "Requirements for high level models supporting design space exploration in model-based systems engineering," in Conference on Systems Engineering Research (CSER 2013), Atlanta, USA, 2013, pp. 293 - 302.
2. B. Kienhuis, E. Deprettere, K. Vissers, and P. v. d. Wolf, "An approach for quantitative analysis of application-specic dataflow architectures," Application-Specic Systems, Architectures, and Processors (ASAP), July, 1997.
3. P. D. Borches, "A3 Architecture overviews," Doctoral Thesis, Department of Engineering Technology (CTW), University of Twente, Enschede, The Netherlands, 2010.
4. B. Wiulsrød, G. Muller, and M. Pennotti, "Architecting Diesel Engine Control System using A3 Architecture Overview," in INCOSE 2012, Rome, 2011.
5. G. M. Bonnema, "Thinking Tracks for Integrated Systems Design," in 1st Joint International Symposium on System-Integrated Intelligence 2012: New Challenges for Product and Production Engineering, Hannover, Germany, 2012.
6. M. Gries, Methods for Evaluating and Covering the Design Space During Early Design Development, UCB/ERL M03/32, EECS Department, University of California, Berkeley, 2003.
7. F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Paserone, and A. Sangiovanni-Vincentelli., "Metropolis: an integrated electronic system design environment," IEEE Computer, vol. 36, no. 4, pp. 45-52, 2003.
8. O. d. Weck, C. Eckert, and J. Clarkson, "A classification of uncertainty for early product and system design," in ICED '07 - International Conference On Engineering Design, Paris, France, 2007.
9. N. Esfahani, K. Razavi, and S. Malek, "Dealing with uncertainty in early software architecture," in Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, Cary, North Carolina, 2012, pp. 1-4.
10. J. Noppen, P. v. d. Broek, and M. Akit, "Software development with imperfect information," Soft Comput., vol. 12, no. 1, pp. 3-28, 2007.
11. M. T. Ionita, "Scenario-Based System Architecting," Doctoral Thesis, Department of Mathematics and Computer Science, Technical University of Eindhoven, Eindhoven, The Netherlands 2005.
12. A. Ivanovic, and P. America, "Customer value in architecture decision making," in Proceedings of the 4th European conference on Software architecture, Copenhagen, Denmark, 2010, pp. 263-278.
13. G. Muller, "CAFCR: A Multi-view Method for Embedded Systems Architecting; Balancing Genericity and Specificity," Technical University of Delft, Delft, 2011.
14. A. Koziolek, "Architecture-driven quality requirements prioritization," in First IEEE International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks), Chicago, IL, 2012, pp. 15-19.
15. G. M. Bonnema, "Funkey architecting: an integrated approach to system architecting using functions, key drivers and system budgets," Doctoral Thesis, University Of Twente, Enschede, 2008.
16. G. S. Altshuller, 40 Principles - TRIZ Keys to Technical Innovation, , Technical Innovation Center, Worcester, MA., 1997.
17. W. P. M. H. Heemels, L. J. Somers, P. van den Bosch, Z. Yuan, B. van der Wijst, A. van den Brand et al., "The Use of the Key Driver Technique in the Design of Copiers," in ICSSEA 2006, Paris, 2006.
18. B. D. Theelen, O. Florescu, M. C. W. Geilen, J. Huang, P. H. A. van der Putten, and J. P. M. Voeten, "Software/Hardware Engineering with the Parallel Object-Oriented Specification Language." pp. 139 - 148, 2007.