Towards Optimization of Design Decisions for Embedded Systems by Exploiting Dependency Relationships

Matthias Riebisch, Alexander Pacholik, Stephan Bode

Ilmenau University of Technology, Germany
{matthias.riebisch | alexander.pacholik | stephan.bode}@tu-ilmenau.de

Abstract: Design decisions for the development of embedded systems demand for a consideration of complex goals and constraints. In order to reduce risks and optimize the design, model-based approaches are needed for an explicit representation of goals and constraints as well as for early assessments. The explicit representation of dependencies is required to make design decisions in a reasonable way. Existing works do not sufficiently support the mapping between problem space and solution space together with a consideration of technological constraints. In this paper the Goal Solution Scheme approach developed for software architectural design is extended for the development of embedded systems considering specific needs for flexible decisions late in the development process. The adaptation of the approach for the relevant goals and development steps of embedded systems is illustrated by its application in a case study of a complex embedded system project.

1 Introduction

During the last years the complexity of embedded software systems has steadily increased. Embedded systems have to satisfy numerous goals simultaneously. This situation results from the systems' integration into heterogeneous environments – regarding both technology and organization. They have become more and more critical for the success of products and services. Furthermore, there is a constant need for optimization and change, together with a high pressure for cost reduction. For example, cost aspects could demand for a change from a hardware- to a software-implementation of a feature at a later point in the development process. To manage complexity and risk, and to provide the required flexibility for late changes of implementation decisions, model-based approaches have been introduced. Model-based development processes help to reduce the risks and to increase the efficiency by providing support with methods and tools.

Decisions during the design process play a key role for the satisfaction of the various goals. Unfortunately, there are competing or even conflicting goals. For optimization, all relevant goals have to be satisfied and balanced. However, method and tool support does not cover all types of goals. Furthermore, there are complex dependencies between the decisions, which limit the set of possible alternatives for a decision – the so-called decision space. The missing comprehension of the dependencies hampers decision-making. To solve this problem, an explicit modelling of these dependencies can provide a base for both effective tool support and the developer's comprehension of the decision space. The modelled information has to cover:

- Goals and preferences. Especially quality goals have to be covered because they can hardly to be achieved by later changes of the implementation.
- Constraints. In the case of embedded systems, a high number of constraints have to be met by solution instruments. For example, certain hardware is not supported by a model-based platform. These constraints restrict the decision space.
- Solution instruments. The potential elements of a solution – even partly abstract ones such as patterns and heuristics – as well as process patterns such as refactorings have to be represented with preconditions for their application and with their impact on goals.

Due to the high risk of the design decisions, a goal-oriented, iterative development procedure together with early assessments is required. Model-based approaches enable such assessments and help to minimize the risks. Other means for risk reduction are (a) simulation – especially for the goals performance and computing power – and (b) prediction – especially for performance and reliability.

The contribution of this paper consists in a model-based, goal-oriented approach, which uses dependency relationships to represent a mapping between problem space and solution space. In this paper the formerly introduced Goal Solution Scheme [Bo09] is adapted to the embedded systems domain. For this adaptation, new mechanisms for the selection of a solution and for decision-making – namely preconditions and constraints – are introduced, together with additional goals and solution instruments. For illustration, the application of the scheme is shown by an example from a large-scale case study.

2 State of the Art

Modelling techniques for competing requirements and goals together with their refinement and resolution have been developed in the Goal-Oriented Requirements Engineering, such as the NFR framework with the Softgoal Interdependency Graph, the i* notation, and the standardization as the User Requirements Notation (URN) [CP09]. The strengths of these approaches consist in their support for the elicitation of requirements and their priorities, the discovery of conflicts, the conflict resolution and scoping, and the support for the classification of the goals. However, they insufficiently consider the transition to the solution space, because the impact of solutions on goals as weights is not sufficiently represented. Furthermore, constraints for the applicability of solutions are not covered.

Support for multi-criteria decision-making is provided by various approaches developed for economy [Tr00]. Several approaches apply a decision matrix to visualize and compare criteria and options, similar to the House of Quality matrix of the Quality Function Deployment method [BR95]. The strength of these approaches consists in the various ways of visualization, which provide support for a manual selection by weighting different factors. Unfortunately, the support for a classification of solution alternatives regarding the goals is rather limited, and the consideration of the transition to the solution space by representing the impact of solutions on goals is missing. Furthermore, these approaches are developed to assist human decision-making, and they do not sufficiently support a tool-based or even model-based one.

Support for mapping between problem and solution space is provided to some extent by all design methodologies. Strongly related approaches in the field of software architectural design are QASAR [Bo00] and ADD [BK02]. They do not sufficiently support the establishment of solutions and the explicit representation of dependencies to goals and constraints.

A classification of solutions is provided by various catalogue approaches, such as Design Pattern catalogues [GH95],[BM96] or the various component catalogues. Unfortunately, most of them do not provide a classification regarding goals. Furthermore, a representation of preconditions for the applicability of tool-based preselection is missing.

For the development of embedded systems, system-level synthesis approaches have been developed. They require as input an executable specification, constraints, and target platform templates; and they apply design space exploration and synthesis to derive ‘optimally’ suited hardware architecture and functional deployment [St10]. However, design decisions on this base require a complete tool chain for synthesis together with platform template databases. Incomplete models and uncertainties in the design prevent the application of such techniques.

A broad literature base provides principles and solution instruments as contributions to system development. The principles for performance optimization in embedded systems design [Wa05] constitute an example. The approach presented in this paper extends these works with a classification regarding goals, which enables their inclusion into layer III and IV of the Goal Solution Scheme (see section 4).

3 Problem Statement

To support design decisions for embedded and software intensive systems, the model-based approach has to fulfil the following objectives:

- **Represent the design space and manage its complexity**. Decisions in this domain have to fulfil complex requirements and various mutual constraints. The set of possible solutions is influenced by methods and principles from different research fields.
- **Manage situations of missing information**. By nature, a design process for a technical system is characterized by incomplete requirements, references to components with yet undefined properties, and missing knowledge about the satisfiability of requirements.
- **Provide support for flexibility**. Due to the mutual constraints and the need for optimization, decisions are wanted to be postponed as far as possible. This covers decisions on platform technology such as hardware and software.

- **Facilitate a goal-oriented evaluation of the solutions**. Due to the high risks and the need for optimization, solutions have to be evaluated as early as possible.

The explicit representation of dependencies in the Goal Solution Scheme, together with its integration in a decision-making process, shall fulfil these objectives.

4 Goal Solution Scheme

For a support of goal-oriented development, a mapping between problem space and solution space is necessary. In iterative development processes, the establishment of decisions and the evaluation of their results are firmly related. The Goal Solution Scheme has been developed to represent a mapping between elements of both spaces by explicit relationships. The layers of the scheme (see Figure 1) correspond to stages of the development process and contain the elements of these stages. Each relation between elements expresses a dependency: a change of one element requires changes of its related elements. A weight added to the relationship expresses its impact, which can be positive or negative. As additional relations, preconditions for the applicability of elements of the solution space are managed, however they are not represented graphically. The layers I and II as well as project constraints represent the problem space, while layers III and IV represent the solution space.

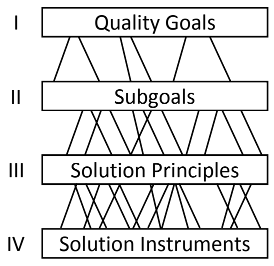

Figure 1: Layers of the Goal Solution Scheme

Layer I covers the top-level goals, such as security, performance, portability, and maintainability – for embedded systems frequently extended by energy consumption, size, and cost. Layer II represents the subgoals. The transition I – II represents a goal refinement, which is derived from quality models, cost models, or similar. The subgoal level facilitates the resolution of trade-off situations between competing goals.

Layer III covers solution principles for a design regarding the different goals. The transition II – III represents the mapping from the problem space to the solution space. The relations represent the impact of a solution principle on the related goals. Examples for positive and negative impact will be discussed in the next section of the paper.

Layer IV contains solution instruments at different levels of abstraction. Examples are building blocks, patterns, reference architectures, and tools for analysis and code generation. The transition III – IV provides a classification of the instruments regarding the principles – and thus regarding the goals. The relations represent the impact of a solution principle on the related goals. In this way, layer IV represents the design space as a set of solutions with properties regarding goals and constraints. The information on layers III and IV as well as the impact relations to the layers above have been acquired and incrementally improved during previous projects [Bo09].

As an extension of the Goal Solution Scheme for the embedded systems domain technological constraints are considered explicitly. It turned out to be necessary to check if the preconditions for the solutions' application are fulfilled. These preconditions are modelled by attributes of solution principles (layer III) and solution instruments (layer IV). They are evaluated during a preselection step by comparison to the constraints of the current design task. Only solution principles and solution instruments with fulfilled preconditions are preselected. In this way the applicable ones are identified. The preselected candidates are then ranked according to their impact on the relevant goals, which is derived from the relationships of the transition II – III – IV of the Goal Solution Scheme. The preselection reduces the number of ranked elements significantly, and thus reduces the complexity of the decision support task.

As a result, a three-step decision-making process is established (see Figure 2), as an extension of earlier works [RW07]. Firstly, the goals are defined based on the preferences of the stakeholders and the constraints and preconditions are identified. Secondly, the constraints and preconditions are compared to preselect a set of candidate solution principles and instruments (layer III and IV). Thirdly, the impact values of the solution principles and instruments and the priorities of the goals are used to calculate weights for the preselected solution instruments to establish a ranking. The resulting ranked lists is then presented to the developer as proposed solution instruments.

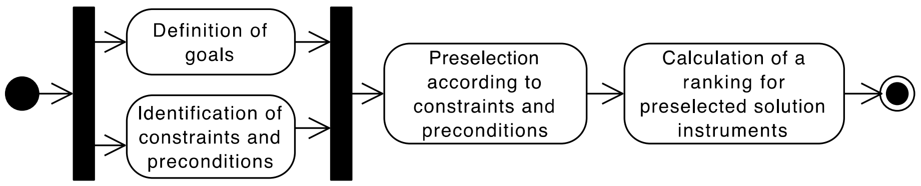
Definition of goals
Identification of constraints and preconditions
Preselection according to constraints and preconditions
Calculation of a ranking for preselected solution instruments

Figure 2: The decision-making process according to the Goal Solution Scheme

The constraints are identified during the requirements analysis or architectural analysis phase. The preconditions for the application of the solution principles and instruments (layer III and IV) are represented by attributes as part of their description within the scheme. For the sake of effort reduction they are partly generalized for classes of solution instruments, as shown in the case study. The preconditions are not visualized.

5 Decision Support

To illustrate the utilization of the Goal Solution Scheme in a design decision, an example from a case study is discussed. The case study deals with the development of a new version of the control unit of a nanopositioning and measuring machine [Mu09]. The purpose and the fields of application of this nanopositioning machine are semiconductor production, biotechnology research, and nano-scale production and research. The objective of the machine consists in a control of the position and the trajectory of an object with a high precision and a high positioning speed. To achieve a high resolution during motions over long distances, classical motion controllers have to be replaced by dynamic control solutions with a higher control loop frequency, which minimize the dynamic control error by estimating the nonlinear dominant disturbance, e.g. friction and stick-slip motion. As a result, the computing requirements increase by orders of magnitude. The production figures for this machine are low, 100 to 200 units per year can be expected.

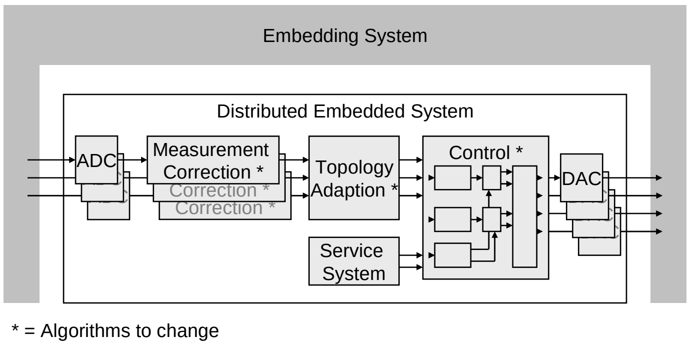

Figure 3: Positioning control unit as an embedded system

The control unit considered here consists of different parts (Figure 3). There are parts for the correction of the deviation of measured input data, for the adaptation of the topology of the measuring and positioning environment, a service system as an interface for user interaction, and the actual controller. The control unit works with blocks for the various control algorithms. There is a demand for a new version of the machine with an increased data volume, speed, and complexity of the control. This requests for a replacement of the formerly used single processor platform by a more powerful one. Furthermore, the processor has to provide floating-point operations. The control algorithms are implemented on the base of models using a tool chain with MATLAB Simulink based code generation [Mu09]. As an alternative, the tool chain supports the derivation of an implementation as programmable hardware via hardware description language HDL.

For an illustration of the approach, the decision on the platform technology for the control unit – implementing the control algorithms – is used as an example. For this decision there are three major alternatives:

1. Software-implementation using a real-time operating system (RTOS) as platform;
2. Software-implementation applying a static scheduling design, without RTOS;
3. Implementation on a heterogeneous platform with programmable hardware, such as FPGA or ASIC.

For both alternatives 1 and 2 there is another choice between a multicore processor and a distributed system as processing platform.

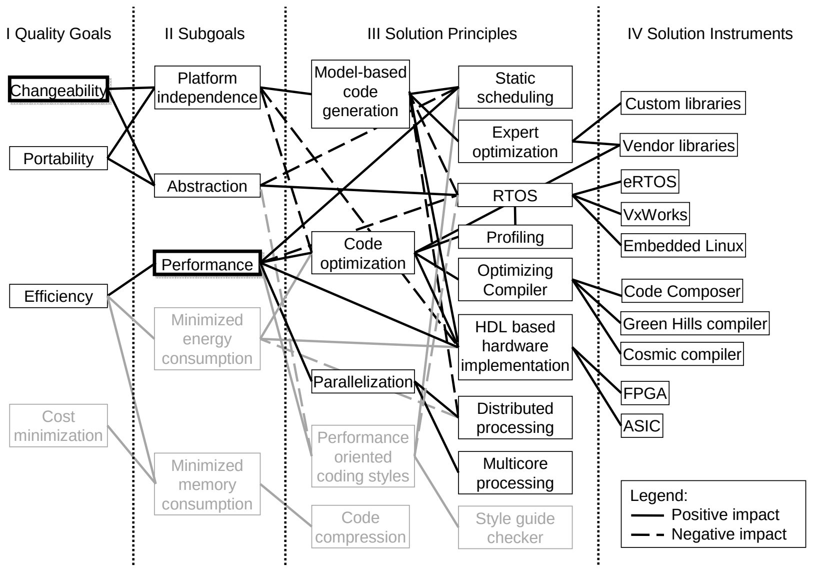

Figure 4: Goal Solution Scheme of the case study (partly)

The goals and their priorities constitute an input for the considered case study. The goals and subgoals for the implementation of the control unit result from the objectives of the nanopositioning and measuring machine:

a. Efficiency regarding *performance*: real time constraints, high positioning speed for a high precision – the most important competitive feature of the system;
b. *Changeability*: the ability to change or replace the positioning algorithms even late in the development process – highly important for the optimization of the precision and speed of the control;
c. *Portability*: exchange of the hardware platform, of the computing platform (processor), and the communication infrastructure – important;
d. Efficiency regarding *minimized energy consumption* of the control unit: need for additional cooling – medium priority, because the control unit could be placed outside.

Goal (a) *performance* got the highest priority by the stakeholders. Goal (b) *changeability* got a high priority, whereas goals (c) and (d) are not discussed further in this paper. Goal (a) is twice as important as goal (b). Therefore, using the well-known analytic hierarchy process, the resulting priorities are 0.66 for goal (a) and 0.33 for goal (b).

During the decision-making, constraints and preconditions are evaluated. Constraints that were identified during requirements analysis and architectural analysis include dependencies between certain development tools, methods, and available hardware.

Preconditions control the preselection. For our example, some of the various preconditions for solution instruments have been generalized to classes (see Table 1). According to the constraints of the project and to those derived from previous decisions, a subset of the available solution instruments is preselected for a further evaluation. We removed the ASIC alternative due to cost and risk. Furthermore we decided to restrict on target hardware and tool chains, which are already available for the development project, more precisely C6000 series floating point DSPs from texas instruments and Virtex V and Spartan III series FPGAs from Xilinx. Therefore only the Code Composer compiler and the eRTOS operating system alternative remain.

Table 1: Preconditions for the application of solution instruments (examples, generalized)

Solution instrument	**Preconditions**
RTOS	Support by selected processor
Compiler	Support for selected processor
Profiling	Support by selected processor
Vendor library	Support for selected compiler
Vendor library	Support for selected processor
FPGA	Support by HDL vendor and tool chain
ASIC	Support by HDL vendor and tool chain
ASIC	Cost: minimum number of units

The next step of the application of the Goal Solution Scheme consists in a consideration of the solution principles (layer III) for the discussed goals. Due to space reasons, only a part of the solution principles is covered by the example (see Figure 4). For the subgoal ***performance*** we consider the solution principles as the ones with a positive impact: *code optimization*, *static scheduling* design, and *parallelization*. Related to them there are more solution principles *optimizing compiler, hardware implementation, distributed* and *multicore processing.* A related solution principle with a negative impact is *RTOS* because of the overhead. The solution principle *performance oriented coding* style is not considered further because the preconditions for its applications are not fulfilled. For the goal ***changeability*** two subgoals *platform independence* and *abstraction* are relevant. For *platform independence*, layer III provides the solution principles *model-based code generation, HDL based hardware implementation,* and – with a negative impact – *code optimization*. For *abstraction*, the solution principles *RTOS* and *static scheduling* (with a negative impact) are the related ones.

There is a trade-off situation between the competing goals (or subgoals) *performance* and *changeability*, which is represented in the Goal Solution Scheme by mutually negative impact relations on layer III (dashed lines in Figure 4). The explicit representation of the modelled dependencies in the scheme facilitates the resolution of this competition during the decision-making. In our example there are two solution principles with a positive impact on both goals, *static scheduling* and *HDL based hardware implementation*. *Model-based code generation* and *HDL based hardware implementation* have a positive impact on *platform independence* and further on *changeability* because of the support of the tool chain. *Static scheduling* is supported by *model-based code generation* due to a derivation of related models by the tool chain.

The impact of *RTOS* to *model-based code generation* is negative because a special support of a certain *RTOS* by code generators is rarely available. Furthermore, the relation between *model-based code generation* and *distributed processing* shows a negative impact because special support for distribution and configuration by code generators is rarely available. These and the other dependencies lead to higher scores of the solution principles *model-based code generation* and *HDL based hardware implementation* compared to other ones such as *RTOS*. This results in a higher ranking of the related solution instruments as proposed solutions for the design decisions.

For the given GSS the impact values between solution instruments and goals have been calculated by backward propagation (for details on the calculation procedure see [BR10]). The resulting matrix contains values between -1 and 1, where the minimum-maximum magnitude ratio is 1:5. Thus the resulting matrix is equivalent to a -5 to +5 evaluation scheme. By multiplying the Matrix with the goal priorities, the final impact values of solution instruments result. The final ranking (best to worst) is as follows: static scheduling, FPGA, multicore processing, distributed processing, vendor libraries, profiling, optimizing compiler, custom libraries, and eRTOS.

6 Conclusion and Future Work

In this paper, decisions for embedded system design are discussed as transitions from problem space to solution space with dependency relationships as links. By an explicit representation of these dependencies in the presented Goal Solution Scheme, the treatment of complex goals, constraints, and dependencies during the decision-making process is simplified. A resolution of competing goals and a simultaneous consideration of multiple design principles and candidate solutions by a developer are facilitated. The application of the Goal Solution Scheme is explained by decision-making examples from a larger case study.

As future work, the extension of the Goal Solution Scheme by more solution principles regarding performance, parallelization, and distributed system design is intended. Furthermore, cost as a major goal shall be included. Related to this goal, principles and procedures for economic decisions have to be analyzed, formulated and incorporated into the layers III and IV of the scheme. Other next steps involve the extension of the tool support for the selection of a broader variety of solution instruments regarding their technological constrains, for example of more real-time operation systems and libraries with their options for profiling and code optimization, and of the various programmable hardware technologies and products with their concrete dependency relations to tool support. With a high coverage of dependencies between the relevant solution instruments, the method of goal-oriented design shall be integrated with the concept of system synthesis [St10] for a certain platform.

7 Acknowledgements

The research on model-based decision methods has been partly funded by the federal state Thuringia and the European Regional Development Fund ERDF through the Thüringer Aufbaubank under grant 2007 FE 9041. The project of the nanopositioning and measuring machine has been partly funded by the German Research Council DFG under grant SFB 622.

8 References

[BK02] Bass, L.J.; Klein, M.; Bachmann, F.: Quality Attribute Design Primitives and the Attribute Driven Design Method. In (F. van der Linden Ed.): Proc. Workshop Software Product-Family Engineering PFE2001, Springer, Berlin, 2002, pp. 169-186.

[Bo00] Bosch, J.: Design and Use of Software Architectures. Addison Wesley, New York, 2000.

[Bo09] Bode, S.; Fischer, A.; Kühnhauser, W.; Riebisch, M.: Software Architectural Design meets Security Engineering. In: Proc. 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS2009), San Francisco, CA, USA, April 13-16. IEEE CS, 2009, pp. 109-118.

[BR10] Bode, S.; Riebisch, M.: Impact Evaluation for Quality-Oriented Architectural Decisions Regarding Evolvability. In: M.A. Babar and I. Gorton (Eds.): 4th European Conference on Software Architecture (ECSA2010), LNCS 6285, Springer, 2010, pp. 182-197

[BR95] Barnett, W.D.; Raja, M.K.: Application of QFD to the software development process. International Journal of Quality & Reliability Management, Vol. 12, No 6, 1995, pp.24–42

[BM96] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.: Pattern-Oriented Software Architecture, Volume 1: A System of Patterns. Wiley. 1996.

[CP09] Chung, L.; do Prado Leite, J.: On Non-Functional Requirements in Software Engineering. In Borgida, A. et al. (Eds.): Conceptual Modeling – Foundations and Applications. Springer, Berlin, 2009, pp. 363–379.

[GH95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley, 1995

[Ma10] MATLAB and Simulink, The Mathworks, Inc. http://www.mathworks.com, 3 Dec 2010

[Mu09] Müller, M.; Amthor, A.; Fengler, W.; Ament, C.: Model-driven Development and Multi-processor Implementation of a Dynamic Control Algorithm for Nanomeasuring Machines. In: Journal of System and Control Engineering (JSCE), Vol. 223, No 3, 2009, pp. 417-429, Prof. Engin. Pub., London.

[RW07] Riebisch, M.; Wohlfarth, S.: Introducing Impact Analysis for Architectural Decisions. Proc. ECBS2007, IEEE CS, 2007, pp. 381-390.

[St10] Streubühr, M.; Gladigau, J.; Haubelt C.; Teich, J.: Efficient Approximately-Timed Performance Modeling for Architectural Exploration of MPSoCs. In Borrione D. (Ed.): Advances in Design Methods from Modeling Languages for Embedded Systems and SoC's, LNEE 63. pp. 59-72, Springer Netherlands, 2010.

[Tr00] Triantaphyllou, E.: Multi-Criteria Decision Making: A Comparative Study. Kluwer, Dordrecht, 2000.

[Wa05] Wasson, C.S.: System Analysis, Design, and Development: Concepts, Principles, and Practices. Wiley, 2005.