An Aspect-Oriented Model-Driven Engineering Approach for Distributed Embedded Real-Time Systems[*]

Marco Aurélio Wehrmeister[1,3], Carlos Eduardo Pereira[1,2], Franz Josef Rammig[4]

[1] Programa de Pós Graduação em Computação (PPGC) – Instituto de Informática
[2] Departamento de Engenharia Elétrica
Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[3]Universidade do Estado de Santa Catarina (UDESC)
Campus Univ. Prof. Avelino Marcante s/n – 89223-100 – Joinville – SC – Brazil

[4]Heinz Nixdorf Institut – Universität Paderborn
Fürstenallee 11 – 33102 – Paderborn – Germany

mawehrmeister@inf.ufrgs.br, cpereira@ece.ufrgs.br, franz@upb.de

Abstract. *To cope with the increasing design complexity of modern embedded real-time systems, Model-Driven Engineering (MDE) techniques are being proposed and applied within the domain of embedded real-time systems. This thesis proposes a design approach combining MDE and aspect-oriented concepts to deal with functional and non-functional requirements in a modularized way using higher abstraction levels. A configurable tool for code generation that supports the proposed methodology is presented. It is capable of creating source code for different target platforms from the models produced in earlier design phases. In addition to code generation for functional requirements handling, the tool also weaves aspects' adaptations, which modify the generated code to handle non-functional requirements. The proposed approach has been successfully applied to the development of embedded real-time systems for different real-world applications. Obtained results show an improvement concerning the modularization of system's requirements handling, leading to an increased reuse of previously created artifacts.*

1. Introduction

The design of embedded systems is not a trivial task. Frequently, designers must deal with software and hardware elements that must be highly optimized to cope with system's requirements. This situation increases the design complexity. Many embedded system researchers and designers propose to raise the abstraction level during design to manage this complexity. In this context, the Object-Oriented (OO) paradigm appears as an interesting option, since several object-oriented approaches have been proposed in the literature to the embedded real-time systems design. However, the domain of embedded real-time systems presents many specific requirements (e.g. deadlines for tasks accomplishment, energy consumption, reduced footprint, communication latency, etc.) that do not specify system's functionalities but are tightly related to them, the so-called non-functional requirements. Traditional approaches (e.g. OO or structured analysis methods) do not have specific constructions to deal with these requirements, whose handling is usually found intermixed with the handling of functional requirements. This situation brings problems such as tangled and scattered handling, which hinder the reuse of previously developed artifacts (e.g.

[*] This thesis was carried out in co-supervision by the doctoral programs of UFRGS (Brazil) and Uni Paderborn (Germany), respectively *Programa de Pós-Graduação em Computação* and *Promotionsausschuss des Instituts für Informatik*. The main supervisor is prof. Dr.-Ing. Pereira, and the co-supervisor is prof. Dr. rer. nat. Rammig. Marco's doctor title is valid in both countries. The complete list of publications is available in `http://www.inf.ufrgs.br/~mawehrmeister/phd_publications.html`.

model elements or code). To address the above-mentioned problems, some proposals can be found in the literature, such as subject-oriented [Ossler and Tarr] and Aspect-Oriented (AO) programming [Kiczales et al. 1997], which provide special constructions to specify and encapsulate the handling of non-functional requirements into single elements.

However, to address these problems only at implementation level is not sufficient to manage embedded real-time systems' design complexity. The increase of the abstraction level is an old but widely accepted idea to help with such quest. Following this claim, embedded systems community is looking for new techniques/approaches, such as Model-Driven Engineering (MDE) [Selic 2003], aiming at the management of design complexity, as it can be seen in a recent embedded systems market survey[1]. An important issue to allow the use of MDE in embedded systems design is tool support [Selic 2003]. Automatic transformation from Platform Independent Models (PIM) to Platform Specific Models (PSM) is a key issue to make models the main artifacts during the whole development cycle instead of source code. Additionally, it avoids errors coming from manual transformations, and also helps to keep specification and implementation synchronized. Code generation from high-level models can be seen as a transformation of PIM into PSM, but instead of using meta-model to meta-model transformations (i.e. transforming meta-model elements from a PIM into PSM meta-model elements), it applies the translation from meta-model to text representing source code in a target language.

This paper presents the *Aspect-oriented Model-Driven Engineering for Real-Time systems* (AMoDE-RT) design approach for embedded real-time system. It combines MDE techniques with AO concepts to address the problems mentioned above by increasing the abstraction level during design, and allowing a modular handling of functional and non-functional requirements. Especially, AMoDE-RT separates requirements handling from earlier design phases until the implementation. For that, it uses an AO requirements analysis method, and allows the specification of aspects within UML models [OMG 008a] and in code generation rules. AMoDE-RT is supported by a code generation and aspects weaving tool called *Generation of Embedded Real-Time Code based on Aspects* (GenERTiCA). This tool takes as input an UML model enhanced with aspects and a set of mapping rules to generate source code for different target platforms. AMoDE-RT has been successfully applied in different real-world applications of embedded real-time systems, focusing on automation applications. Obtained results show remarkable improvements concerning modularization of system's requirements handling. As a consequence, previously developed artifacts (e.g. model elements, mapping rules, code) have been easily reused. These results, along with the increase in the abstraction level (by means of using UML models), lead to evidences that a decrease in design complexity has been obtained, since designers do not need to concern on low level details in new designs using the same target platform.

This paper is structured as follows: Section 2 provides an overview on the AMoDE-RT approach; Section 3 discusses the aspect-oriented framework proposed in this work; Section 4 describes GenERTiCA's main features; Section 5 presents the validation of this work, and finally Section 6 discusses some conclusions and future work directions.

2. Related Work

This section provides a brief assessment of related works, highlighting the main contributions of this thesis. A more comprehensive evaluation of the state-of-art is provided in [Wehrmeister 2009]. Concerning MDE techniques for embedded and real-time systems, approaches such as [Edwards and Green 2003] and [Perseil and Pautet 2008] use

[1]`http://www.embedded.com/design/testissue/210200580`

UML with its standard semantics, i.e. using OO concepts that do not deal with functional and non-functional requirements in a modularized and effective way. On the other hand, approaches such as [Tesanovic et al. 2005] and [Rajkumar 2007] provide separation of concerns to handle requirements by using AO concepts, but do not use a standard modeling language, i.e. they propose their own modeling language. This work proposes to merge both worlds, i.e. it combines AO concepts with UML-based MDE to design embedded real-time systems. It proposes high-level aspects to handle a subset of non-functional requirements commonly found in embedded and real-time systems domains, allowing separation of concerns during the whole development cycle.

In addition, there are many code generation tools. [Andersson and Höst 2008] proposes a tool to generate SystemC code from state diagrams. Commercial tools, e.g. Rational Rose [IBM 2010] or Artisan Studio [Artisan 2010], can generate classes' skeleton code from class diagrams for Java and C/C++ languages. These tools usually do not combine structural and behavioral information specified in different UML's diagrams, and hence, only generate automaticaly a minor code portion. Normally, these tools are not flexible since their generation process cannot be customized to support different target languages. GenERTiCA proposes a more innovative approach: (i) its script-based approach allows to generate code for different target platforms; (ii) it separates concerns on the specification of transformation rules; (iii) it uses UML's structural and behavioral diagrams (not only class or state diagrams) to generate code; and (iv) it considers aspects (specified in the UML model) during code generation, allowing the use of AO concepts in non-AO languages.

3. Overview of AMoDE-RT

Aspect-oriented Model-Driven Engineering for Real-Time systems (AMoDE-RT) [Wehrmeister et al. 2007] allows a smooth transition from initial specification phases to implementation phases. By using MDE techniques combined with AO concepts, AMoDE-RT increases the abstraction level during design to address the increasing complexity of embedded real-time systems. Figure 1 shows an overview of AMoDE-RT.

The first step in AMoDE-RT is gathering requirements and constraints of the embedded real-time system. This is performed using the RT-FRIDA [Freitas et al. 2007] approach, which provides a methodology and toolset (e.g. templates, checklists, mapping tables, and requirements conflict resolution tables) to assist in gathering functional and non-functional requirements. The discussion of RT-FRIDA is out of scope of this paper, interested readers should refer to [Freitas et al. 2007].

The next step is the modeling phase. The requirements gathered in the previous step are used to specify system's structure and behavior using UML [OMG 008a] diagrams annotated with stereotypes of the MARTE profile [OMG 008b]. These diagrams are successively refined up to achieving the desired level of detail, providing sufficient in-

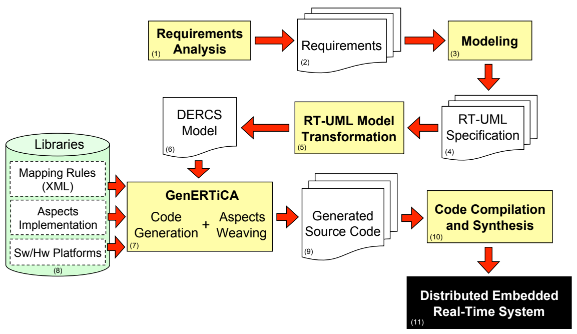

Requirements Analysis (1)
Requirements (2)
Modeling (3)
RT-UML Specification (4)
RT-UML Model Transformation (5)
DERCS Model (6)
Libraries
Mapping Rules (XML)
Aspects Implementation
Sw/Hw Platforms (8)
GenERTiCA
Code Generation + Aspects Weaving (7)
Generated Source Code (9)
Code Compilation and Synthesis (10)
Distributed Embedded Real-Time System (11)

Figure 1. Overview of the AMoDE-RT design approach

formation for system realization. In the initial UML model, elements describe concepts that are closer to the target application domain (e.g. sensors, steering devices, robot arms, etc.).Higher abstraction levels are easier to understand and allow designers to focus on applications foundations instead of concerning about implementation issues. These elements represent the handling of functional requirements. The specification of non-functional requirements is carried on using a high-level aspects framework called *Distributed Embedded Real-time Aspects Framework* (DERAF) [Wehrmeister 2009]. DERAF is used in two moments: (i) earlier design phases and (ii) in the implementation phase, more specifically, in the code generation/aspects weaving step. For details see Section 4.

Although increasing the abstraction level during design is good for managing complexity, the higher the abstraction level is, more are the chances of ambiguous or erroneous interpretations of the same specification. As UML allows specification of overlapping information (i.e. the same feature can be specified with different diagrams and/or elements), the produced model can be ambiguous. However, for code generation purposes, these ambiguities must be removed. These issues are addressed as follows: (i) modeling guidelines; and (ii) model transformation. Considering (i), a set of guidelines has been defined in AMoDE-RT, i.e. a subset of UML diagrams is used, as well as rules defining how to specify application's elements using these diagrams. Considering (ii), transformation rules have been created to transform UML models into instances of the *Distributed Embedded Real-time Compact Specification* (DERCS) [Wehrmeister et al. 2009] , a more concise PIM suitable for code generation (see Section 5). UML-to-DERCS transformation is performed automatically by GenERTiCA tool. Unfortunately, due to space constraints, this paper cannot detail these issues. Interested readers are a referred to [Wehrmeister 2009].

The third and fourth steps of AMoDE-RT (boxes 5 and 7 in Figure 1) are performed by GenERTiCA. Source code generation is performed from the DERCS model, which has been created from the UML model. In fact, GenERTiCA performs not only code generation, but also aspects weaving. The code generation process executes a set of small scripts with mapping rules, in order to perform model-to-text transformations from DERCS elements to constructions in the target platform. These scripts create code fragments. Source code files are made up of these generated fragments. The code generation process iterates all DERCS model elements looking for the script that defines the mapping from the element into suitable constructs in the target platform. Furthermore, if the element under evaluation is affected by any DERAF aspect, GenERTiCA weaves the aspects adaptations into the generated code fragment. GenERTiCA uses DERAF aspects implementations, which are also scripts containing target platform's constructions, to modify the code fragment. More details on GenERTiCA are provided in Section 5

Finally, third party tools are used to compile and synthesize the generated application code. Thereafter, system's implementation is almost ready to be executed or tested. Details on all subjects presented in this Section are provided in [Wehrmeister 2009].

4. Aspects Framework for Modeling and Implementation

One of the main contributions of this work is the *Distributed Embedded Real-time Aspects Framework* (DERAF). DERAF provides a set of aspect to deal with a subset of the non-functional requirements [Freitas et al. 2007] found in the domain of embedded real-time systems. DERAF is an extensible high-level aspects framework to be used in both earlier design and implementation phases. The main idea is to provide aspects that enhance the modeled system by means of adding specific behavior and structure to specify non-functional requirements handling. These "new" features are independent from any specific implementation technology. Designers choose DERAF aspects to specify the non-functional requirements handling based on aspects' high-level semantics, which must be followed in their implementation using a target platform. Thus, in UML model, DERAF aspects are used as "black boxes". Some examples of DERAF aspects are: *PeriodicTiming* provides all behavior to deal with periodic activation of tasks; *ConcurrentAccessControl* provides all features to control the access of shared resources; and *EnergyMonitoring* adds elements to monitor energy consumption, warning if a specified threshold is reached.

DERAF aspects are specified in UML using the *Aspects Crosscutting Overview Diagram* (ACOD) [Wehrmeister et al. 2007], an extension to the class diagram proposed in this work. In addition to ACOD, this work also proposes a new profile to specify AO concepts in UML models. ACOD shows aspects and classes (which are affected by them). Aspects are depicted as classes annotated with the `<<aspect>>` stereotype, along with information on their *adaptations*, *join points*, and *pointcuts*. Join points describe the selection of which elements are affected by aspects' adaptations. For their specification, this work uses *Join Point Designation Diagrams* (JPDD) [Stein et al. 2006].

As previously stated, DERAF aspects are also used during code generation process. Aspects' adaptations are implemented as scripts that are executed by GenERTiCA. In other words, a script contains target platform's constructions that implement aspects' semantics. For instance, an adaptation script can append new commands at the end of a generated code fragment. This extra code deals with the non-functional requirement(s) addressed by the DERAF aspect. As demonstrated in the developed case studies, it is possible to use AO concepts with non-AO target languages, such as Java or C++.

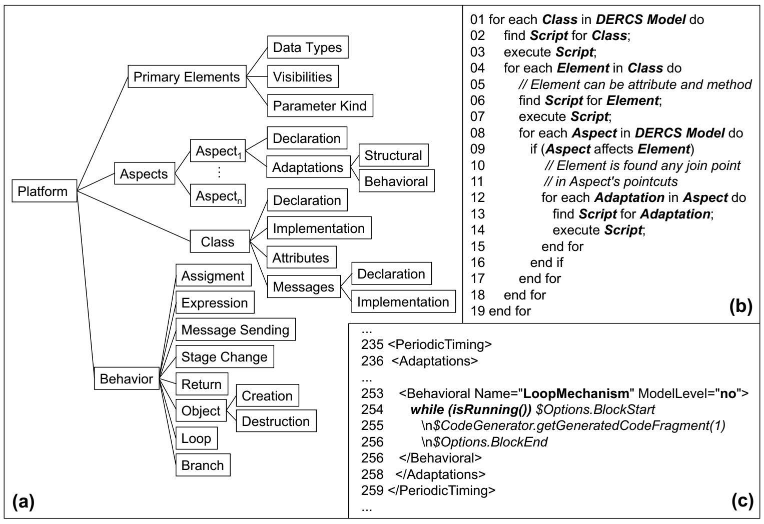
5. Code Generation and Aspects Weaving Tool

According to [Selic 2003], to enable the use of MDE, it is important to provide tool support any proposed MDE approach. Thus, to support AMoDE-RT, this work created the *Generation of Embedded Real-Time Code based on Aspects* (GenERTiCA) [Wehrmeister et al. 2008], a configurable tool for code generation and aspects weaving. It automates the production of source code from UML models. For that, GenERTiCA transforms a UML model into a DERCS model, upon which mapping rules are applied to generate code and also to weave aspects adaptations.

As mentioned, the *Distributed Embedded Real-time Compact Specification* (DERCS) has been proposed to address the ambiguity problem in UML models, aiming at source code generation. There are two remarkable contributions of DERCS: (i) it combines behavior information which are spread in many different diagrams' elements of UML's meta-model into fewer elements; and (ii) it provides meta-model elements to represent AO concepts. Basically, a DERCS model represents functional requirements handling in terms of OO concepts (e.g. objects, methods' behavior, etc.), and non-functional requirements handling using DERCS' AO elements (e.g. aspects, pointcuts, join points, etc.).

Additionally, this work proposed UML-to-DERCS transformation heuristics, which are followed by GenERTiCA to keep the equivalence between models. For instance, a sequence diagram can produce different DERCS's `Behavior` elements (with their sequence of actions) depending on the diagram's message nesting. Furthermore, sequence diagrams annotated with `<<JPDD>>` stereotype are transformed into DERCS' `JoinPoint` elements. JPDDs are evaluated and all DERCS' elements that match with their search criteria are selected and included in the corresponding `JoinPoint` elements. There are many more transformation rules, for details see [Wehrmeister 2009].

GenERTiCA is a script-based code generation tool. It uses a set of small scripts



```
for each Class in DERCS Model do
    find Script for Class;
    execute Script;
    for each Element in Class do
        // Element can be attribute and method
        find Script for Element;
        execute Script;
        for each Aspect in DERCS Model do
            if (Aspect affects Element)
                // Element is found any join point
                // in Aspect's pointcuts
                for each Adaptation in Aspect do
                    find Script for Adaptation;
                    execute Script;
                end for
            end if
        end for
    end for
end for
```

```
...
<PeriodicTiming>
 <Adaptations>
...
   <Behavioral Name="LoopMechanism" ModelLevel="no">
     while (isRunning()) $Options.BlockStart
        \n$CodeGenerator.getGeneratedCodeFragment(1)
        \n$Options.BlockEnd
   </Behavioral>
  </Adaptations>
</PeriodicTiming>
...
```

Figure 2. (a) Mapping rules; (b) Code generation/aspects weaving process; (c) Example of an aspect's Implementation

(written using the VTL[2]) to produce code fragments from the DERCS model. There is a script for each DERCS' element. Thus, GenERTiCA does not use a single script to specify mapping rules for all model's elements, as it is commonly found in code generation tools. In this sense, GenERTiCA's approach improves the separation of concerns to define mapping rules from model elements into target platform's constructions, since designers need to focus only on few elements per script rather than the whole meta-model.

GenERTiCA's scripts are organized in a XML file according to the organization depicted in Figure 2a. Each node represents a DERCS element. Scripts are stored in tree's leafs. The code generation and aspects weaving process is outlined in Figure 2b. GenERTiCA passes through all DERCS elements, trying to match the element under evaluation with any XML tree's node, in order to find the appropriate mapping rule script for that element. Once the script is found, it is executed and the code fragment is generated. A script has complete access to element's information, as well as to the entire DERCS model, allowing the specification of both simple and elaborated mapping rules. Although it was not formally proved, this code generation approach preserve the equivalence between model's elements and target language's constructions, i.e. by comparing the generated code with the mapping rules scripts, it is possible to trace back to the source element in the model.

As mentioned, DERAF aspects' implementations are also done using script. Figure 2c shows a small example. As depicted in Figure 2b, if the element under evaluation is affected by any aspect, its adaptations are executed and the generated code fragment is modified to include aspects' non-functional requirements handling. GenERTiCA's aspect weaving approach, and also the use of DERAF in UML model, are remarkable contributions, since they allow designers to use AO concepts even if the target platform does not support AO. Although the simplicity of this approach, it separates effectivelly the requirements handling, as demonstrated in the case studies performed in the scope of this work.

6. Validation and Results

The proposed ideas were validated using three real-world applications of embedded real-time systems that were used as case studies: the movement control system of an unmanned aerial vehicle; the control system of an industrial packing system; and, the movement control of an automated wheelchair. For each case study, two versions have been cre-

[2]Velocity Template Language, `http://velocity.apache.org`

ated: one object-oriented and one aspect-oriented.They have been compared using a subset of the software engineering metrics for AO systems presented in [Sant'anna et al. 2003]. For these case studies' implementation, mapping rules for two different target platforms, namely RT-FemtoJava [Wehrmeister et al. 2004] and ORCOS platforms [UPB 2008], have been specified to generate source code from their AO version's model.

The case studies helped on evaluating the reusability quality of the artifacts created during design, and also the decrease in the effort to produce systems' implementation. Reusability was assessed by means of understandability and flexibility factors. They have internal attributes such as system size, cohesion, coupling and separation of concerns, which can be measured by the mentioned AO metrics. To summarize, almost all metrics (calculated upon the case studies' models) have better values for AO model compared to OO one, ranging from 37% to 66% in average. Considering the understandability factor, key issues such as separation of concerns, cohesion and coupling improved around 45% in average. For flexibility factor, AO model elements are more cohesive and decoupled compared to OO model. Separations of concerns results show that elements in AO model have more specific and well-defined roles than in OO model.

Further, DERAF aspects have been reused in different projects. The case studies used 12 from a total of 21 aspects available in DERAF (i.e. 57%). From them, 8 aspects (66%) were used in at least two case studies. Furthermore, if the implementation follows aspects adaptations' high-level semantics, their implementation can also be reused, as occurred in all case studies. AMoDE-RT's approach for join points specification also allows JPDDs reuse, since 52% of all created JPDDs were used in all case studies.

Regarding GenERTiCA usage, the generated source code files are more complete[3] than the ones obtained using available commercial or academic code generation tools, which usually only provide class skeletons and/or simple state machine code. In these case studies, it was possible to generate an amount of source code lines from 1.73 to 4.2 times the amount of mapping rules script's lines. For instance for RT-FemtoJava, from a total of 338 script lines, 3080 lines of Java was generated (an average amount of 1026 lines per case study). It is importat to highlight that the mapping rules were create in the first case study and reused without modification for the other two. A comprehensive discussion on this assessment is provided in [Wehrmeister 2009]

7. Conclusion and Future Work

This paper presented a design approach to deal with the increasing design complexity of embedded real-time systems. A design is better understood if its functional and non-functional concerns are well separated. Artifacts can be reused in different designs with less effort if they are cohesive and decoupled. Thus, it is expected that previously developed artifacts can be easily reused to decrease the effort and shorten the design time. In this sense, AMoDE-RT can be considered a step towards this aim, since it provides not only separation of concerns in earlier design phases, but also a smooth transition from requirements specification to source code implementation.

Case studies' results leads to the conclusion that improvements achieved with AMoDE-RT increase with the number of crosscutting non-functional requirements. Extracted metrics confirm that, using MDE and AO, the same benefits achieved in traditional information systems can also be obtained in the design of embedded real-time systems. In this sense, DERAF is a remarkable contribution due to the lack of aspects with platform independent semantics created specifically to embedded real-time systems domain.

[3]GenERTiCA has generated not only classes' attributes and methods signatures, but also their associated behavior

Considering the code generation, it has been demonstrated that GenERTiCA produces a greater amount of source code from UML models. GenERTiCA is able to generate not only classes' structure (i.e. attributes and methods signature), but also their associated behavior. Other contribution is that GenERTiCA considers aspects adaptations during code generation process, enabling the use of AO concepts even in non-AO platforms.

As future work, other case studies are being executed. Mapping rules are being created for other target platforms, e.g. VHDL. In addition, the presented approach is intended to guide the development of an adaptable and flexible middleware for wireless sensor networks. AO concepts may allow the inclusion of middleware's different features as they are demanded, and this way, promoting the desired flexibility and adaptability.

References

Andersson, P. and Höst, M. (2008). Uml and systemc: Comparison and mapping rules for automatic code generation. In Villar, E., editor, *Embedded Systems Specification and Design Languages*, pages 199–209.

Artisan (2010). Real-Time Studio. `http://www.artisansoftwaretools.com`.

Edwards, M. and Green, P. (2003). Uml for hardware and software object modeling. In Lavagno, L., Martin, G., and Selic, B., editors, *UML for Real: Design of Embedded Real-Time Systems*, pages 127–147.

Freitas, E. P. et al. (2007). DERAF: A high-level aspects framework for distributed embedded real-time systems design. In *Early Aspects: Current Challenges and Future Directions*, pages 55–74. Springer.

IBM (2010). Rational Rose. `www-01.ibm.com/software/awdtools/developer/rose`.

Kiczales, G. et al. (1997). Aspect-oriented programming. In *Proc. of European Conference on Object-Oriented Programming*, pages 220–242, Berlin. Springer-Verlag.

OMG (2008a). Unified Modeling Language. `www.omg.org/spec/UML/2.2/Superstructure`.

OMG (2008b). UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE). `www.omg.org/cgi-bin/doc?ptc/2008-06-08`.

Ossler, H. and Tarr, P. Using subject-oriented programming to overcome common problems in object-oriented software development/evolution. In *Intl. Conf. on Software Engineering*, pages 687–688.

Perseil, I. and Pautet, L. (2008). Foundations of a new software engineering method for real-time systems. *Innovations in Systems and Software Engineering*, 4(3):195–202.

Rajkumar, R. (2007). Model-based development of embedded systems: The sysweaver approach. In Ramesh, S. and Sampath, P., editors, *Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems*, pages 35–46. Springer Netherlands.

Sant'anna, C. et al. (2003). On the reuse and maintenance of aspect-oriented software: An assessment framework. In *Anais do XVII Simpósio Brazilerio de Engenharia de Software*, number 17, pages 19–24.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE Software*, 20(5):19–25.

Stein, D. et al. (2006). Expressing different conceptual models of join point selections in aspect-oriented design. In *5th Intl. Conf. on Aspect-Oriented Software Development*, pages 15–26, New York. ACM.

Tesanovic, A. et al. (2005). Aspects and components in real-time system development: Towards reconfigurable and reusable software. *Journal of Embedded Computing*, 1(1/2005):17–37.

UPB (2008). Organic reconfigurable operating system. `http://orcos.cs.uni-paderborn.de`.

Wehrmeister, M. A. (2009). *An Aspect-Oriented Model-Driven Engineering Approach for Distributed Embedded Real-Time Systems*. PhD thesis, Federal University of Rio Grande do Sul, Brazil. `www.inf.ufrgs.br/~mawehrmeister/wehrmeister_thesis_final.pdf`.

Wehrmeister, M. A. et al. (2004). Optimizing real-time embedded systems development using a rtsj-based api. In *On the Move to Meaningful Internet Systems: OTM 2004 Workshops*, pages 292–302. Springer.

Wehrmeister, M. A. et al. (2007). An aspect-oriented approach for dealing with non-functional requirements in a model-driven development of distributed embedded real-time systems. In *10th Intl. Symp.on Object Oriented Real-Time Distributed Computing*, pages 428–432, Washington. IEEE Computer Society.

Wehrmeister, M. A. et al. (2008). GenERTiCA: A tool for code generation and aspects weaving. In *11th IEEE Intl. Symp. on Object Oriented Real-Time Computing*, pages 44–54. IEEE Computer Society.

Wehrmeister, M. A. et al. (2009). An infrastructure for UML-based code generation tools. In *Analysis, Architectures and Modelling of Embedded Systems*, volume 310/2009, pages 32–43. Springer, Boston.