Using Aspects to Model Distributed Real-Time Embedded Systems

Edison Pignaton Freitas[1], Marco Aurélio Wehrmeister[1], Elias Teodoro Silva Jr.[1], Fabiano Costa Carvalho[1], Carlos Eduardo Pereira[1,2], Flávio Rech Wagner[1]

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
[2] Dep. Engenharia Elétrica – Universidade Federal do Rio Grande do Sul (UFRGS)

{epfreitas, mawehrmeister, etsilvajr, fccarvalho, flavio}@inf.ufrgs.br, cpereira@ece.ufrgs.br

Abstract. *Distributed Real-time Embedded (DRE) systems have several requirements directly related to characteristics that are difficult to handle when a pure object-oriented method is used for their development. These requirements are called non-functional requirements and refer to orthogonal properties, conditions, and restrictions that are spread out over the system. Pure object-oriented methods do not address successfully those concerns, so new technologies, like aspect orientation, are being applied in order to fulfil this gap. This work presents a proposal to use aspect orientation in the analysis and design of DRE systems. To support our proposal, we performed an adaptation of a well-defined method called FRIDA (From RequIrements to Design using Aspects), which was originally applied to the fault-tolerant domain. The proposed adaptation includes the use of RT-UML together with aspect-oriented concepts in design phase, aiming to separate the handling of non-functional from functional requirements.*

Resumo. *Sistemas de Tempo-real Embarcados e Distribuídos (TrED) apresentam diversos requisitos de difícil tratamento quando se utiliza uma metodologia orientada a objetos para sua modelagem. Estes requisitos são classificados como não-funcionais e referem-se à propriedades ortogonais, condições e restrições espalhadas por todo o sistema. Metodologias orientadas a objetos não cobrem satisfatoriamente tais requisitos, o que motiva a aplicação de novas tecnologias, como a orientação a aspectos, para cobrir esta lacuna. Este trabalho apresenta uma proposta de uso de orientação a aspectos para a análise e projeto de sistemas TrED. Para isto, adaptamos uma metodologia chamada FRIDA (From RequIrements to Design using Aspects), que foi originalmente aplicada ao domínio de sistemas tolerantes a falhas. A adaptação proposta inclui o uso de UML-RT juntamente com conceitos da orientação a aspectos na fase de projeto, visando a separação do tratamentos dos requisitos não-funcionais dos funcionais.*

1. Introduction

The increasing complexity of distributed real-time embedded (DRE) systems requires new techniques to improve the design in order to allow the system evolution, maintainability, and reuse of previously developed artifacts. Nowadays, an important concern involved in DRE system design is how to deal with non-functional requirements (NFR).

The main concept when dealing with NFR is the crosscutting concerns, which really worries system developers. If not properly handled, those concerns are responsible for tangled code and loss of cohesion. In the literature, it is possible to find several works addressing this separation of concerns where the crosscutting concerns are identified as NFR, as in [Chung and Nixon 1995]. In order to handle the separation of concerns, several works propose guidelines to handle NFR separately from the functional ones. Among those works stand out subject-oriented programming [Ossher and Tarr 1999] and aspect-oriented programming [Kiczales 1997]. Both approaches address the problem at the implementation level, but the development community realized that the NFR must be taken into account as soon as possible to enhance the system design. This fact motivates pushing the separation of concerns to the early phases of the design, as in the Early-Aspects [Rashid *et al* 2002] approach.

Real-time systems have a very important NFR, which is the concern about the time in the execution of their functionality. The complexity related to non-functional analysis increases when those systems become distributed and embedded. To deal with these non-functional requirements, many proposals suggest the use of aspects, as in [Stankovic *et al* 2003].

Our work presents an approach to deal with the complexity exhibited by NFR in DRE systems by adapting the FRIDA (From RequIrement to Design using Aspects) [Bertagnolli and Lisbôa 2003] method to the DRE domain to be used in conjunction with RT-UML [OMG 2004]. This approach emphasizes the separation of concerns from the early phases of system development. The adaptation of the FRIDA toolset to deal with DRE systems concerns enables a clear specification of system requirements, which are easier to map into design elements. Additionally, the use of RT-UML together with aspect-oriented elements is an interesting option in the design phase. Another noteworthy contribution is the improvement of traceability from requirements to design.

The remainder of this paper is organized as follows. In Section 2, the original FRIDA model is introduced and its adaptation to the DRE domain is presented. A case study is used in Section 3 to illustrate the use of the method. Section 4 discusses related work, while final remarks, conclusions, and future work are presented in Section 5.

2. The FRIDA Model

FRIDA is a well-defined method that offers a sequence of phases to support requirement analysis and system design. The main goal is to deal with the complexity of NFR and to separate them from the functional ones, beginning from the early phases of system life cycle. The method is based on aspect-oriented software development.

Considering NFR in system analysis is a way of avoiding code tangling and the undesired mixing of different concerns in later phases, which is always present in systems developed with current object-oriented (OO) methods. The problem with the OO paradigm is that it is simply unaware of NFR, on other words, there is no specific element dedicated to handle NFR. OO considers the system just in the functional dimension, without special concerns with NFR. Using an AO approach, FRIDA tries to fill this gap by providing a way to consider both functional and non-functional requirements, handling them with a special focus on the non-functional ones. The

method was developed in the scope of reliable fault-tolerant systems, but its tools are flexible enough to be adapted to other domains.

FRIDA is divided in six main phases. Each phase is connected to each other in a way to provide traceability among project elements and system requirements. Figure 1 presents the whole method. The first phase is dedicated to identifying the system functional requirements. Use case diagrams and templates are used to elicit those requirements. In the second phase, the non-functional requirements are identified and specified. To perform this task, check-lists, lexicons, and conflict resolution rules are used. A link among classes, actors, and the use cases elicited in the first phase is created in the third phase. The next phase performs almost the same, but for the non-functional requirements, representing them visually in the class diagram. In the fifth phase, the functional requirements of the project, represented by classes, are linked to aspects. Finally, in the last phase, the source code of classes and aspect skeletons is generated.

2.1. The FRIDA Model Applied to the DRE Domain

FRIDA provides a consistent method to separate non-functional from functional requirements from the early phases of system development, representing a relevant contribution to the system analysis and to the mapping of requirements into design elements (for details see [Bertagnoli and Lisboa, 2003]).

FRIDA focuses on the fault tolerance domain, with a vocabulary and tools designed to support the analysis of fault tolerant systems. In order to adapt FRIDA to the DRE domain, the first step was to identify the concerns related to DRE development. Some key requirements of this domain are shown in Figure 2. Those requirements are based mainly on the study present in [BURNS 1997] and in the IEEE glossary [IEEE 2006]. Based on this classification, some FRIDA tools were adapted to consider those requirements. It is important to highlight that many DRE systems also have fault tolerance requirements. Considering these issues, every requirement considered in the original FRIDA model can also be used in the development of a DRE system.

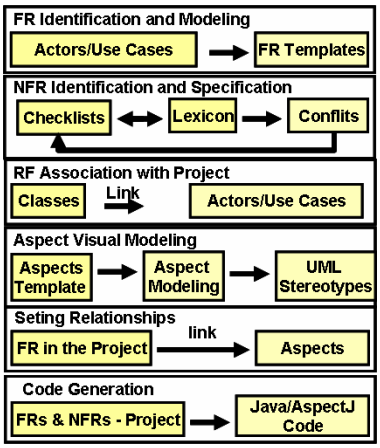

Figure 1. Original FRIDA Method [Bertagnolli and Lisbôa 2003]

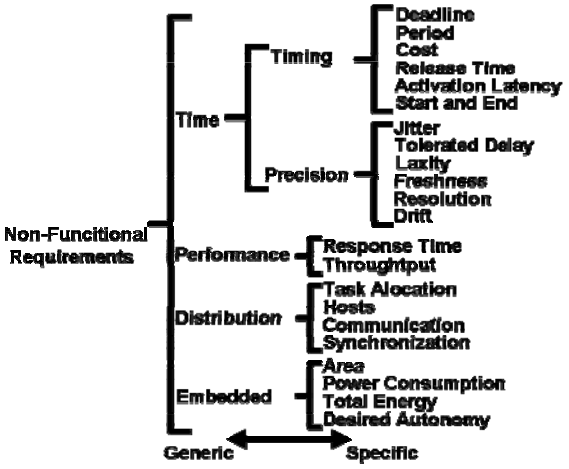

Figure 2. NFRs classification for DRE Systems

In order to explain the use of the adapted method, a case study showing the automation of a wheelchair is considered. The following section describes the case study and each phase of the adapted method.

3. Case Study

The case study consists of the design of a distributed real-time embedded automation and control system for an "intelligent" wheelchair to support people with special needs. Hard real-time requirements (e.g. comply the deadline of collision detection task) must be accomplished for safety reasons. The whole automation project includes functions like movement control, collision detection, automatic movement, scheduled movement (e.g. convey patient to room 11 at 10:00 am). In this paper we will concentrate our attention in requirements that are related to the movement control.

The wheelchair can be controlled manually through a joystick or automatically by pre-defined movements (or by a route) stored in its data base. In both cases, the movement control system has to monitor the movement in order to avoid collisions and prevent any system malfunction. In this case, a corrective action must be taken in a short time.

3.1. Requirements Identification and Specification

This initial phase consists of phases one and two of the original method. The analysis starts with the identification of the functional requirements to build the use case diagram. The next step is to fulfill the templates that specify each identified use case.

At this point, the analysis handles the NFR, using a set of check-lists in order to elicit the NFR present in the system. Four check-lists for the DRE domain have been developed, covering the following areas: time, performance, distribution, and embedded. Each check-list can have sub-check-lists describing how specific and how generic is the requirement. For instance, a question that appears in the check-list of "embedded" concerns regards the power consumption constraints like system autonomy. An example of a full check-list for "timing" concern is presented in Figure 3(a). The first column lists the non-functional requirements and the inferring questions, while the second one means relevancy of the requirement, the third column gives its priority and fourth column gives information about restrictions, conditions, and/or a description of the requirement. Additionally, other check-lists for each generic NFR presented in Figure 2 were created. Due to space restrictions, only one check-list is presented.

After the use of check-lists, it may happen that a given NFR could not be satisfactorily specified or even could not be identified at all. To refine the identification and specification of NFRs, a lexicon is used. This lexicon consists of rules organized in *Backus Naur Form* (BNF) [Naur, Backus 1969]. An example of a lexicon for timing requirements can be seen in Figure 3(b). As it was done for check-lists, specific lexicons for each generic NFR were also created.

After the identification of NFRs, the next step is to identify conflicts among them, and thus a matrix with all identified non-functional requirements is built. If a requirement conflicts with another one, the cell in the matrix that meets both requirements is checked signalizing the conflict. The priorities defined in the check-lists are used to solve the identified conflicts. If two or more requirements that are in conflict have the same priority, the stakeholders must be consulted in order to decide which is more important.

	Rel	Pr	R/C/D
Time			
Timing			
Is there any periodic activity or data sampling?	**X**	8	Joystick data read Movement Control & Sensoring
Is there any sporadic activities?			
Is there any aperiodic activity?			
Is there any restriction in relation to the latency to start an execution of a system activity?	**X**	9	Corrective Action
Is there any specific instant to start or finish an execution of a system activity?			
Precision			
Is there flexibility in the time requirements of any activity?	**X**	10	No, they must be respected
Is it acceptable the existence of a delay in any timed activity?			
Is it acceptable any variation in the time requirements?	**X**	8	Control variables not affected
Is it possible to use old (not fresh) data?			
Is there a limit in the drift of the logic time in relation to the physic time?	**X**	10	drift< 5 milliseconds

Figure 3a. Check-list example

<NFR_generic> ::= <time> | <performance > | <distribution> | <embedded>

<time> ::= <timing> | <precision >

<timing> ::= <deadline> | <period> | <cost> | <release_time> | <activation_latency> | <start_end>

<deadline> ::= an execution must be done until <n> <time_unit>

<period> ::= each <n> <time_unit>

<cost> ::= consume <n> <time_unit>

<release_time> ::= an activity must be ready to execute in <n> <time_unit>

<activation_latency> ::= after released, an activity must execute in <n> <time_unit>

<start_end> ::= an activity starts in <n> <time_unit>| an activity finishes in <n> <time_unit>

<time_unit> ::= h | min | s | ms | μs | ns | hour | minute | second | millisecond | microsecond | nanosecond | day | week | month | year

<n> ::= <n> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 3b. Lexicon example

After the conflicts elimination and also the decision about which NFR will be considered in the system development, the next step is to fulfill a template for each NFR, describing their features. The NFR template is shown in Figure 4 with an example of its use. The column "Item" describes the evaluated NFR feature. The column "Description" gives the meaning of each entry and the "Case Study" column gives an NFR example from the wheelchair case study. This example shows the specification of the "periodicity" feature of the "timing" NFR.

The final step in this phase is to complete the use case diagram with the considered NFR. As stated before, this paper focuses on the movement control subsystem. All the expected functionality of the wheelchair movement control is shown in Figure 5. As can be observed, some functions have non-functional aspects affecting their behavior. Those non-functional requirements affect different functions, which will certainly imply a decentralized handling in the final system (making harder the reuse and maintainability). In this case study, we consider concerns about timing and distribution. The first one has two facets: the timing control that handles the execution of the activities, and the timing parameters that handle all information about time constraints. The second one deals with the distribution problem, in this case specifically with the synchronization that must exist in the concurrent accesses to data that are stored in elements that run in different nodes.

	Item	Description	Case Study
Identification	**Identifier**	An identification that will allow the traceability of the concern over the whole project.	NFR-1
	Name	Crosscutting concern's name.	Timing (Periodicity)
	Author	The responsible for the concern identification and definition.	Edison Pignaton de Freitas
Specification	**Classification**	Class to which the concern belongs.	Time/Timing/Period
	Description	Description of how the concern affects system functionalities.	The system has some activities that must be executed in regular periods of time. These activities are: (1) data acquisition from joystick; (2) data acquisition from movement sensors; and (3) the control of wheelchair movement.
	Affected Use Cases	List of the use cases affected by the concern.	(1) Joystick Sensoring; (2) Movement Sensoring; (3) Wheelchair Movement Control
	Context	Determines when the concern is expected to affect a use case.	Each time that a new cycle of data reading or movement controlling starts.
	Scope	(Global/Partial) The requirement is global if it affects the whole system, and it is partial if affects only a part of the system.	Global
Decision and Evolution	**Priority**	A number used to decide the relative importance among non-functional concerns.	8
	Status	A requirement can have one of the following status: 0 - identified; 1 - analysed; 2 - specified; 3 - approved; 4 - cancelled; 5 - finished;	5

Figure 4. The template used to specify NFR

The notation used in this work is not standardized by the OMG. It follows ideas taken mainly from [Araújo et al 2002] and [Stein et al 2002]. In Figure 5, it can be seen how the non-functional requirements explained above affect the desired system functionality, represented with the same syntax of a use case with a stereotype applied over it («non-functional»), indicating that this is a crosscutting concern. In order to represent how those non-functional requirements crosscut system functions, an arrow goes from the element representing the non-functional requirements to the affected use case. This arrow is noted with the stereotype «crosscut».

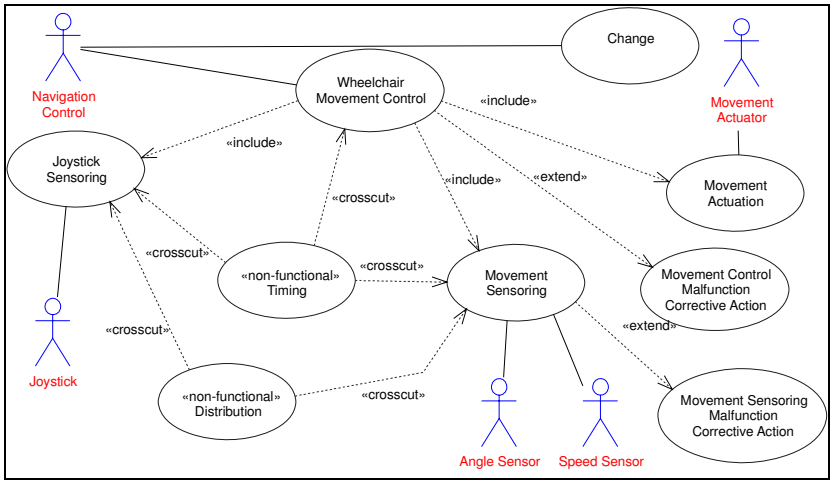

Figure 5. The use case diagram of the wheelchair movement control with NFR.

3.2. Requirement Association with Project Elements

In this phase, the designer maps the requirements (identified and specified in the first phase) with elements that take part in the system project. There are three main tasks that have to be performed in this phase:

- Extract from the use case diagram and FR templates the concepts and attributes that will compose the system functional part. This functional part consists of classes that will be detailed in the next phase, where a class diagram is created and populated;
- Extract the aspects from the information contained in the use case diagram and the NFR templates. This information will define the aspects that will handle each identified and specified NFR;
- Composition of previously extracted information into a mapping table that will link the requirements with the project elements. This table is very important to guarantee the traceability of requirements over the system life cycle. Additionally, this table relates the functional requirements with non-functional ones that affect them. Figure 6 shows the organization of the mapping table.

		Non-Functional Requirements				
		Timing	Distribution	...	NFR n	**Classes responsible for handling FRs**
Functional Requirements	Joystick Sensoring	**X**	**X**			JoystickDriver JoystickInformation
	Movement Control	**X**		**X**		MovementController MovementActuator
	…				**...**	...
	FR n		**X**			Class n
Aspects responsible for handling NFRs		Timing	TaskSynchronization	...	Aspect n	

Figure 6. The mapping table relating FRs to classes and NFRs to aspects.

The mapping table shown in Figure 6 is organized as follows: non-function requirements are set in the top row; function requirements are set in the first column from the left side. Aspects that handle a specific NFR are set in the bottom row in the corresponding column, while classes that handle a FR are set in the column in the right side of the table, in the corresponding row. Cells relating FR that are affected by NFR are marked with an "X". It is important to highlight that as well as some FR can be handled by more than one class, NFR can also be handled by more than one aspect.

3.3. System Design

The structure of the system can be built using the information collected, analyzed, and organized in the previous phases. The class diagram is used to represent system structure and, as stated above, is populated using the information contained in the mapping table. Additionally, details of each requirement can be found in the templates filled in the first phase. Figure 7 presents a class diagram with the modeled elements of the movement control subsystem.

The first step for building this diagram is to represent the class hierarchy of the system. The UML profile for Schedulability, Performance and Time Specification (RT-UML) [OMG 2004] is used to annotate classes that represent a schedulable resource (i.e. an active object that has its own executing thread) or another remarkable characteristic that has to be highlighted in the model (e.g. access policy and resource usage of classes that represent shared objects, which are accessed concurrently by active objects). One of the major novelties in our approach is to combine RT-UML stereotypes with aspects in design phase, taking advantage of a well-defined modeling standard to real-time domain raising the abstraction level in design by using aspects.

The next step is to represent the NFRs with the respective aspects handling them. These aspects are represented as classes annotated with the «aspect» stereotype. One aspect can improve an element by adding new elements (introduction) or by modifying existing ones (advice). A query expression (for advices, this query is called a *pointcut*) identifies which elements are affected by the aspect. In the diagram of Figure 7, the "Timing" aspect affects active object classes (i.e. those classes annotated with «SAschedRes», that is, the schedulable resources), by adding new attributes and setting them up. Additionally, the "Timing" aspect improves active objects that have periodic execution (i.e. those classes that have a message annotated with «SAtrigger» and the tagged value "RTat=periodic"), by implementing the activation frequency control of the active object execution.

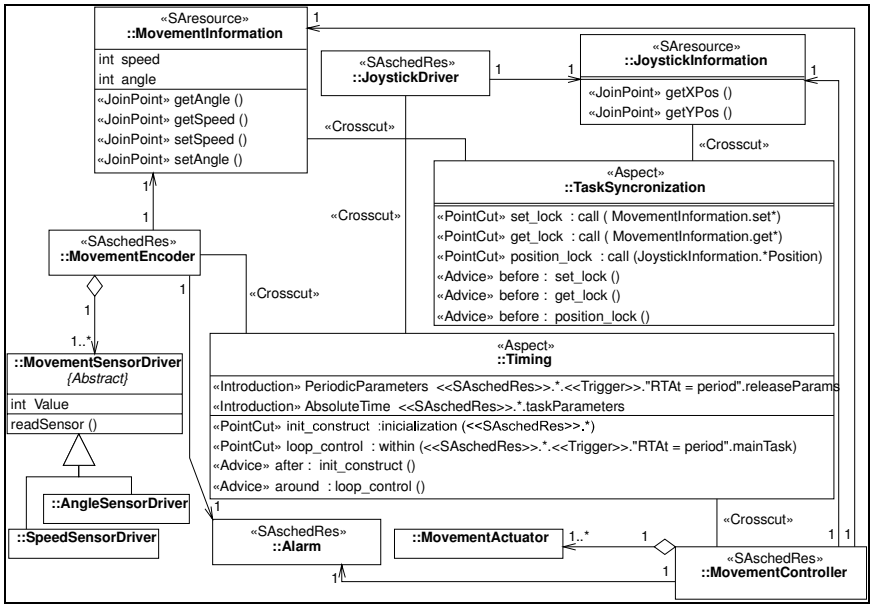

Figure 7. Class diagram populated with classes and aspects.

The final step is to connect the aspects with the affected classes. This information can also be taken from the mapping table by the checked cells in the table. This relation is characterized by a line connecting the aspect with those affected classes, annotated with the stereotype «crosscut».

Observing Figure 7, it is possible to notice that all NFR handling related to real-time and distribution concerns were encapsulated in separate model elements. In current OO methods, those concerns would be spread out over several elements in the model

(e.g. attributes and object behaviours). The separation of concerns in aspects and classes makes easier to find and change the handling of those concerns.

4. Related Work

Even if aspect-orientation is a relatively new concept, there are some proposals to use it in DRE systems, especially to handle real-time requirements. The majority of the works in this domain propose the use of aspects in the implementation phase, like the approach presented in [Tsang *et al* 2004].

Another remarkable work in the area is [Stankovic *et al* 2003]. This work proposes a set of tools named VEST (Virginia Embedded System Toolkit) that uses aspects to compose a new DRE system based on a component library. Those aspects check the possibility of composing components with the information taken from system models. Results presented in the paper depict the design time reduction to build a DRE systems using VEST. This work uses the concept of aspects to check and test dependencies among library components. We propose a different approach in which aspects are used to directly model non-functional requirements since the analysis phase.

Some proposals bring the concept of aspects to early phases of a system development, like [Araújo *et al* 2002] and [Zhang and Liu 2005]. Those proposals had a strong influence in the present work. The first one proposes the use of aspects in requirements analysis and its notation in UML use cases. Another interesting feature of this proposal is the use of templates to describe NFRs. The second proposal describes a way to separate functional and non-functional requirements in the system structure. This is done by the use of stereotypes to represent aspects in class diagrams. In the present work, both ideas are used in addition to the concepts and tools presented in FRIDA. However, an important advantage of the current work is the use of RT-UML together with aspects. This composition brings together the well-defined elements from RT-UML and the separation of concerns supported by aspects.

5. Conclusions and Future Work

This paper proposes the use of aspect-orientation to develop high quality Distributed Real-time Embedded systems using an adapted version of the FRIDA method. By adapting a well-defined method to the DRE domain, like FRIDA, the goal is to provide efficient tools to analyze and model non-functional requirements of this domain in a clear way. This separation of concerns from early phases of development allows a better understanding of system complexity and also a better base to build the system structure. Another advantage is the improvement of reuse of system components, because the non-functional handling is not intermixed in functional elements. The use of RT-UML is advantageous because it enables the application of knowledge from the real-time community, which is materialized as a UML profile.

This work does not present a code generation phase as the original FRIDA method. There is an ongoing work that will provide a code generation tool, however this tool will not follow the original FRIDA proposal, which creates only code skeletons. The idea, instead, is to generate source code as complete as possible using aspect- and object-oriented information provided by UML models. However, the programming language used to generate the code will not be an AO language, instead of this, it will be

an OO or even a procedural language (depending on the mapping from model to code) with the modeled aspects woven in the generated source code. On other words, the tool will weave aspects and generate code from the UML model of DRE application. Additionally, we plan to incorporate the adapted FRIDA method into a greater project, named SEEP (Wehrmeister *et al. 2005*), to incorporate aspect-oriented concepts in the design method proposed into the referred project.

References

Araújo, J., Moreira, A., Brito, I., Rashid, A. (2002) "Aspect-Oriented Requirements with UML", Workshop on Aspect-oriented Modeling with UML, UML 2002, Dresden, Germany.

Bertagnolli, S. C., Lisbôa, M. L. B. (2003) "The FRIDA Model", In: Analysis Aspect-Oriented Software, Germany, (Held in conjunction with ECOOP 2003).

Burns, A., Wellings, A. (1997), Real-time systems and programming languages, Addison-Wesley, 2nd edition.

Chung, L. and Nixon, B.A. (1995) "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach", In: Proc. of 17th International Conference on Software Engineering, ACM Press, pp. 25 – 37.

Institute of Electrical and Electronics Engineering (2006), "IEEE Standard Glossary", http://standards.ieee.org/catalog/olis/arch_se.html

Kiczales, G., et al., (1997) "Aspect-Oriented Programming", In: Proc. of European Conference for Object-Oriented Programming, ECOOP, Lecture Notes in Computer Science 1241, Springer-Verlag, pp. 220-240.

Naur, P., Backus, J. W. et al, (1969) "Revised Report n the algorithmic Language Algol 60", Programming Systems and Languages, Edited by Saul Rosen, New York, McGraw-Hill.

OMG - Object Management Group. (2004), "UML Profile for Schedulability, Performance, and Time Specification", http://www.omg.org/cgi-bin/doc?ptc/04-02-01, February 2004.

Ossler, H., Tarr, P. (1999) "Using subject-oriented programming to overcome common problems in object-oriented software development/evolution", In: Proc. of 21st International Conference on Software Engineering, IEEE Computer Society Press, pp. 687-688

Rashid, A., Sawyer, P., Moreira, A., Araujo, J. (2002) "Early Aspects: A Model for Aspect-Oriented Requirements Engineering", In: Proc. of IEEE Joint International Conference on Requirements Engineering, pp. 199-202.

Stankovic, J. A. et al., (2003) "VEST: An Aspect-Based Composition Tool for Real-Time System", In: Proc. of 9th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS, pp. 58-59.

Stein, D., Hanenberg, S., Unland, R. (2002) "A UML-based Aspect-Oriented Design Notation for AspectJ", In: Proc. of International Conference on Aspect-Oriented Software Development, pp.106-112.

Tsang, S. L., Clarke, S., Baniassad, E. (2004) "An Evaluation of Aspect-Oriented Programming for Java-based Real-Time Systems Development", In: Proc. of the 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'04).

Wehrmeister, M.A., Becker, L.B., Wagner, F.R., Pereira, C.E. (2005) "On Object-Oriented Platform-based Design Process for Embedded Real-Time Systems", In Proceedings of the 8th IEEE (ISORC'05).

Zhang L., Liu, R. (2005) "Aspect-Oriented Real-Time System Modeling Method Based on UML". In Proc. 11. IEEE – RTAS 05.