USING ASPECT-ORIENTED CONCEPTS IN THE REQUIREMENTS ANALYSIS OF DISTRIBUTED REAL-TIME EMBEDDED SYSTEMS

Edison P. Freitas[1], Marco A. Wehrmeister[1,3], Carlos E. Pereira[1,2], Flavio R. Wagner[1], Elias T. Silva Jr[1], Fabiano C. Carvalho[1]

[1]*Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brazil*
{epfreitas, mawehrmeister, flavio, etsilvajr, fccarvalho}@inf.ufrgs.br

[2]*Departamento de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Brazil*
{cpereira}@eletro.ufrgs.br

[3]*Heinz Nixdorf Institute, University of Paderborn, Germany*

Abstract: Distributed Real-time Embedded (DRE) systems commonly have several requirements 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. In general, the specification of those requirements using pure object oriented methods leads to intermixed specification with the functional requirements. This work presents a proposal to use the concepts of aspect orientation in the specification of DRE requirements at the system analysis phase, offering a link from those requirements to system elements in the design phase. To support our proposal, it was performed an adaptation of a method called FRIDA (From RequIrements to Design using Aspects) to the DRE generating the RT-FRIDA (Real-Time FRIDA).

Key words: Requirements Specification, Distributed Real-time Embedded Systems, Aspect-Orientation, Separation of Concerns.

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. The complexity in developing DRE systems appears since the early phases, when the requirements are being specified. Much of the problem in this specification is related to the handling of the non-functional requirements (NFR) and their traceability in the following design phases.

What really worries system developers when dealing with NFR is the crosscutting concerns which characterize the main concept related to the problem of handling NFR. If not properly handled, those concerns are responsible for tangling code and leak of cohesion. In the literature, there are several works addressing this subject, such as [1][6]. In order to handle the separation of concerns, several works propose guidelines to separate NFR from the functional ones. Among those works stand out subject-oriented programming [11] and aspect-oriented programming [8]. Both approaches address the problem at the implementation level, however the handling of 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 [12] approach.

Real-time systems have very important NFR, which is the concern about the predictability in the execution. The complexity related to NFR analysis increases when those systems become distributed and embedded. To deal with these NFR, many proposals suggest the use of aspects [13][15].

This paper presents an approach to deal with the complexity exhibited by NFR in DRE systems by adapting the FRIDA [2] method to the DRE domain, allowing a clear specification of the system requirements, which can be easily mapped into design elements. This paper focuses the requirement specification in the analysis phase of DRE systems.

The remainder of this paper is organized as follows. Section 2 presents the original FRIDA and RT-FRIDA. Section 3 show the use of RT-FRIDA method through a case study. The related work is discussed in section 4. Finally, the final remarks and future work are presented in Section 5.

2. THE FRIDA MODEL

FRIDA is a 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 separately from the functional ones. The method is based on concepts of the Aspect-Oriented (AO) paradigm [8].

Considering NFR in system analysis is a way of avoiding code tangling and the undesired mixing of different concerns in later design phases, which is always present in systems developed with object-oriented (OO) methods.

The problem with the OO paradigm is that there is no specific element dedicated to handle NFR because only the functional dimension of the system is considered. Using an AO approach, FRIDA tries to fulfill this gap by providing a way to consider both functional and NFR during the specification of systems.

FRIDA is divided in six main phases. Fig. 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 NFRs 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 is created in the third phase. The next phase performs almost the same, but for the NFR, representing them visually in the class diagram. In the fifth phase, the functional requirements are represented by classes that are linked to aspects. Finally, in the last phase, the source code of classes and aspect (i.e. skeletons for classes and aspects) is generated.

2.1 RT-FRIDA: Aplying FRIDA to the DRE Domain

The FRIDA method provides a consistent way 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 [2].

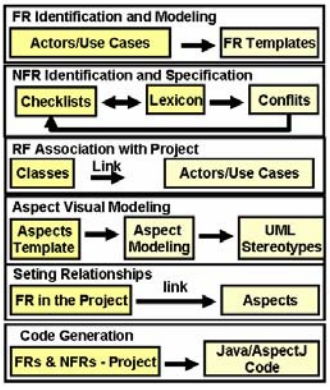

Figure 1. Original FRIDA Method [2]

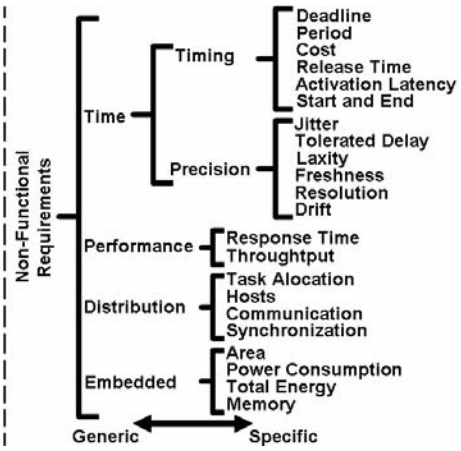

Figure 2. NFRs classification for DRE Systems

FRIDA focuses on the fault tolerance, 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 requirements of this domain are shown in Fig. 2. Those requirements are based mainly on the study present in [3] and in the

IEEE [7] and SEI [5] glossaries. Based on this classification, some FRIDA tools were adapted. It is important to highlight that many DRE systems also have fault tolerance requirements. Thus every requirement considered in the original method can also be used in the RT-FRIDA.

In order to explain the use of the RT-FRIDA, a case study showing the specification of the requirements of a movement control system of an Unmanned Air Vehicle (UAV) is considered. The following section describes the case study and details each phase of the adapted method.

3. CASE STUDY: USING RT-FRIDA

The case study consists on the design of distributed, real-time and embedded movement control system to equip an Unmanned Air Vehicle (UAV). The UAV consists in a helicopter that can be used in a variety of applications, like environment monitoring, support the rescue of victims of natural disasters, urban security, among many other useful applications. The whole design includes functions like navigation, movement control, collision detection, target pursuit, system maintenance, mission management, camera control and data exchange with an on ground base. More details can be consulted in [4]. The presented case study focus on the movement control system, which is composed by sensors and actuators, and also have interface with the maintenance, navigation and data transfer systems. Its operation consists of: (i) the receiving of new movement parameters from navigation system; (ii) the reading of environment and movement sensors; and (iii) the calculus of the guiding and the piloting parameters that will determine the new values that must be applied over rotor actuators. To achieve those operations with success, the system must meet several real-time requirements related to the rate of sampling data, period and deadline of several tasks, among others. The system has several sensors and actuators spatially separated, thus there is also many distribution requirements that must be observed, like communication between processors and concurrency control over shared data. The main concern about the embedded dimension in this system is the energy monitoring and control. As the system can operate in different modes, that depend on the circumstances (i.e. UAV in danger, the “go-back-home” policy establishes the maximum priority to the movement control in spite of the tasks like camera operation), a energy level monitor and control must adequate the system to meet essential requirements, like in hard real-time tasks.

3.1 Requirements Identification and Specification

The analysis starts with the identification of the functional requirements to build the use case diagram. The first 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 created, 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 "embedded" concern is presented in Fig. 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 Fig. 2 were created. Due to space restrictions, only one check-list is presented.

	Rel	Pr	R/C/D
Embedded			
Area			
Is there any restriction in relation to the area used by a system component?			
Power consumption			
Is there any restriction about the power consumption of any system component?	X	10	Consumption optimization according to the operation mode.
Is there a need of controlling the use of the energy in the system?	X	10	Even in a non-special operation mode, the consumption must be monitored and controlled.
Total Energy			
Is any restriction about the available energy?			
Is there an estimate of how long the energy must last?			
Is there any alternative energy source?	X	3	Solar energy can be used.
Memory			
Is there any restriction about data storage?			
Is there restriction about the use of memory to the running code?			

Figure. 3a. Check-list example

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

<embedded> ::= <area> | <power_consumption> | <total_energy> | <memory>

<area> ::= <n> logic cells | <n> <lenght_unit>²

<power_consumption> ::= <n> Watts | <n> Joules by operation

<total_energy> ::= <n> Joules | <n> Joules by component | <autonomy>

<autonomy> ::= <n> of operations | <n> de operations by <time_unit> | <n> <length_unit> covered | <n> <time_unit>

<memory>::= <n> <data_unit> of storage | <n> <data_unit> permanent memory | <n> <data_unit> of non-permanent memory | requires <n> <data_unit>

<length_unit> ::= km | m | dm | cm | mm | kilometer | meter| decimeter | centimeter| millimeter

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

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

<data_unit> ::= bit | byte | Kbit | Kbyte | Mbyte | Tbyte | Megabit | Megabyte | Gigabit | Gigabyte | Terabyte

Figure. 3b. Lexicon example

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 NFR, a lexicon is used. This lexicon

consists of rules organized in *Backus Naur Form* (BNF) [9] that can be used to parse descriptions (in natural language) of the system. An example for embedded requirements can be seen in Fig. 3(b). In the same way that was done for check-lists, specific lexicons for each generic NFR were created.

The next step is to identify conflicts among NFR through the creation of a matrix with all identified NFR. 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 conflicting requirements have the same priority, the stakeholders must be consulted in order to solve them.

After the conflicts removal and the decision about which NFR will be considered in the system development, the next step is to fulfill a template for each NFR. The use of one NFR template is shown in Figure 4. 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 presented UAV case study. This example shows the specification of the "energy" feature of the "embedded" NFR.

	Item	Description	Case Study
Identification	Identifier	An identification that will allow the traceability of the concern over the whole project.	NFR-8
	Name	Crosscutting concern's name.	Power Consumption Monitoring
	Author	The responsible for the concern identification and definition.	Thomas Alva Edison
Specification	Classification	Class to which the concern belongs.	Embedded/Power Consumption
	Description	Description of how the concern affects system functionalities.	The system consumes energy to perform its activities. This consume must be measured by each activity and controlled to achieve an optimized consumption.
	Affected Use Cases	List of the use cases affected by the concern.	(1) Environment Sensing; (2) Rotor Sensing; (3) Piloting; (4) Guidance. (5) Helicopter Movement Control; (6) Signal Alarm
	Context	Determines when the concern is expected to affect a use case.	Each time an activity is performed, the current level of energy must be measured before and after the activity execution.
	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.	10
	Status	Requirement possible status are: 0 - identified; 1 - analysed; 2 - specified; 3 - approved; 4 - cancelled; 5 - finished;	5

Figure 4. The template used to specify 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 system, however all the expected functionality of the UAV movement control is shown in Figure 5. As can be observed, some functions have NFR affecting their behavior. Those NFR 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, precision, distribution and embedded. 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 how imperative is to meet time requirements. The third handles the distribution problem, in this case specifically with the synchronization that must exist in the concurrent accesses to data stored in different nodes, and task allocation over different nodes. The forth is the embedded concern that is related to the energy consumption of the system.

The notation used in this work is not standardized by the OMG. It follows ideas taken mainly from [1]. In Figure 5, it can be seen how the NFR explained above affect the desired system functionality, where each affected functionality (represented through use cases) is annotated with a stereotype that represents the corresponding NFR that affects it. As can be seen in the Figure 5, a number of use cases are affected by several NFR. It is important to highlight that several NFR can affect more than one functionality.

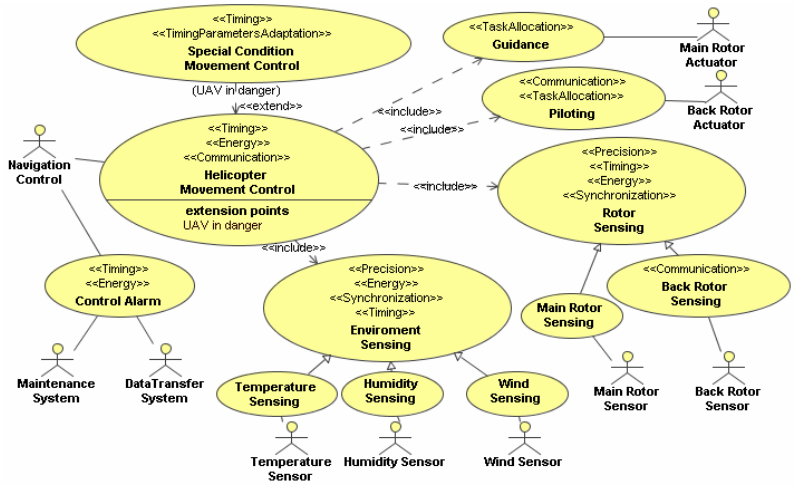

Figure 5. The use case diagram of the UAV movement control with NFR.

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 design. There are three main tasks that have to be performed in this phase:

- Extract from the use case diagram and from the templates of functional requirements the concepts and attributes that will compose the system functional part. This consists of classes that will be detailed in the design phase;

- 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 NFR. It is important to notice that in this phase the developer has an aspect framework called DERAF [16] that contains a set of aspects capable to handle DRE 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.

		Non-Functional Requirements				Classes responsible for handling FRs
		Periodicity	Energy Monitoring	...	NFR n	
Functional Requirements	Helicopter Movement Control	X	X			ControlSubSystem
						MovementControl
	Movement Sensing	X	X	...		MovementSensing SubSystem
						MovementEncoder
						MovementInformation
	...				...	...
	FR n					Class n
Aspects responsible for handling NFRs		Periodic Timing	Energy Monitoring Energy Control	...	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: NFR 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 functional requirements are set in the column in the right side of the table, in the corresponding row. Cells relating functional requirements that are affected by NFR are marked with an “X”. It is important to highlight that as well as some functional requirements can be handled by more than one class, NFR can also be handled by more than one aspect.

3.3 Design Phase

The design phase is not the focus of this paper, but as it is a part of the RT-FRIDA, this section will draw an overview of some issues in this phase. Design phase consists of the construction of diagrams that represent the proposed solution to the system. With the information retrieved from the mapping table and from the templates, the designer can construct the class diagram that represents the structural of the elements that are responsible to handle the functional requirements. The method uses also RT-UML (i.e. UML diagrams annotated with the real-time profile [10]) notation to improve the richness of the used diagram.

In order to represent the non-functional dimension, the method proposes the use of two different diagrams: (i) the Aspect Crosscutting Overview Diagram (ACOD), which shows the aspects affecting classes; and (ii) the Join Point Designation Diagram (JPDD) [14], which emphasizes points in which functional elements (e.g. class, message exchange, etc) are affected by an aspects. More details about the design phase can be seen in [16].

4. RELATED WORK

The aspect-orientated paradigm is a relatively new concept, however 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 [15].

Another remarkable work is [13]. 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. Even being an important work, it does not present a concern about the separation of requirements in the analysis level, but only in the design and implementation.

Some proposals bring the concept of aspects to early phases of a system development, like [1] and [17]. Those proposals had influenced the present work. The first one proposes the use of aspects in requirements analysis and its notation in UML use cases, but does not provide a link with design elements. The second proposal describes a way to separate functional and non-functional requirements in the system structure, but besides of it does not consider the analysis phase, it proposes the use of UML diagrams with much text-based information in tags to explain how to handle NFR.

5. FINAL REMARKS AND FUTURE WORK

This paper proposes the use of aspect-orientation to develop high quality DRE systems using an adapted version of the FRIDA method. By adapting a well-defined aspect-oriented method to the DRE domain, like FRIDA, the goal is to provide efficient tools to analyze and model NFR of this domain. This separation of concerns from early phases of development allows a better understanding of system complexity and also a better base to build its structure. It can improve the reuse of system components, because each NFR can be handled by specific components in the system design, avoiding the intermixed specification of system elements.

As future work, the authors intend to incorporate the RT-FRIDA method into a CASE tool to assist the system development, as well as to assist the

designer to obtain metrics about the quality of the design at earlier stages. Another ongoing activity is the definition and implementation of a framework of aspects (DERAF) to handle NFR during modeling and implementation phases of DRE systems.

REFERENCES

1. Araújo, J., Moreira, A., Brito, I., Rashid, A. "Aspect-Oriented Requirements with UML", Workshop on Aspect-oriented Modeling with UML, UML (2002), Dresden, Germany.
2. Bertagnolli, S. C., Lisbôa, M. L. B. "The FRIDA Model", In: Analysis Aspect-Oriented Software, Germany, (Held in conjunction with ECOOP 2003).
3. Burns, A., Wellings, A. Real-time systems and programming languages, Addison-Wesley, 2nd edition (1997).
4. Cai, G., Peng, K. , Chen, B. M., Lee, T. H. "Design and Assembling of a UAV Helicopter System", Proc. of the International Conference on Control and Automation, (ICCA2005), IEEE Computer Society, (2005), pp.697-702.
5. Carnegie Mellon Software Engineering Institute, "Online Technical Terms Glossary", http://www.sei.cmu.edu/str/ indexes/glossary/, Sep. 2006
6. Chung, L. and Nixon, B.A. "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 (1995).
7. Institute of Electrical and Electronics Engineering (2006), "IEEE Standard Glossary", http://standards.ieee.org/catalog/olis/arch_se.html
8. Kiczales, G., et al., "Aspect-Oriented Programming", In: Proc. of ECOOP, Lecture Notes in Computer Science 1241, Springer-Verlag, pp. 220-240 (1997).
9. Naur, P., Backus, J. W. "Revised Report n the algorithmic Language Algol 60", (1969) Programming Systems and Languages, Edited by Saul Rosen, New York, McGraw-Hill .
10. Object Management Group, "UML profile for Schedulability, Performance and Time", v.1.1, www. omg.org/cgi-bin/doc?formal/2005-01-02, Sep.2006
11. Ossler, H., Tarr, P. (1999) "Using subject-oriented programming to overcome common problems in oo software development/evolution", In: Proc. of 21st International Conference on Software Engineering, IEEE Computer Society Press, pp. 687-688
12. Rashid, A., Sawyer, P., Moreira, A., Araujo, J. "Early Aspects: A Model for Aspect-Oriented Requirements Engineering", In: Proc. of IEEE Joint International Conference on Requirements Engineering, pp. 199-202, (2002).
13. Stankovic, J. A. et al., "VEST: An Aspect-Based Composition Tool for Real-Time System", In: Proc. of 9th IEEE RTAS, pp. 58-59, (2003).
14. Stein, D., Hanenberg, S., Unland, R. "Expressing Different Conceptual Models of Join Point Selections in Aspect-Oriented Design", Proc. of 5th Int. Conf. on Aspect-Oriented Software Development, ACM Press, (2006), pp.15-26
15. Tsang, S. L., Clarke, S., Baniassad, E. "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), (2004).
16. Wehrmeister, M.A., Freitas, E.P., Wagner, F.R., Pereira, C.E. "Applying Aspect-Orientation Concepts in the Model-Driven Design of Distributed Embedded Real-Time Systems", To appear in the Proceedings of the 10th IEEE ISORC'07, (2007).
17. Zhang L., Liu, R. "Aspect-Oriented Real-Time System Modeling Method Based on UML". In Proc. 11. IEEE RTAS'05, (2005).