Goals and Quality Characteristics: Separating Concerns[*]

Elena Navarro
Computer Science Department
University of Castilla-La Mancha
Avda. España S/N, Albacete, Spain
enavarro@info-ab.uclm.es

Patricio Letelier, Isidro Ramos
Department of Information Systems and Computation
Polytechnic University of Valencia
Camino de Vera s/n, Valencia, Spain
{letelier, iramos}@dsic.upv.es

Abstract

Software Requirements Specification (SRS) organization for complex and/or large systems have to do with several not faced challenges until the moment. This organization is a key factor to facilitate the quality assurance of the SRS, regarding features as: correctness, completeness, consistency and modifiability. Organization is also crucial for an effective exploitation of the SRS when elaborating other related or derived artefacts. Although there is a consensus about SRS content, this is not applicable to the organization. Nevertheless, it is evident that depending on the system, their stakeholders and the activities to perform with the SRS, the relevant criteria for SRS organization and presentation can be different. Additionally, another of the main problems to be solved is related to the crosscutting of requirements that produces tangled specifications. This work faces these issues: the organization of SRS by applying Aspect Oriented techniques to properly manage the crosscutting. A Goal Oriented approach for requirements allows us to establish traceability from software goals to specific requirements and from the latter to operationalizations that are realized as software components. In this work, we present an integration of aspect and Goal Oriented approaches, to properly manage the SRS organization and presentation. Furthermore, our proposal uses the standard ISO/IEC 9126 as the starting point to organize goals and requirements. ATRIUM, a methodology to concurrently define requirements and software architecture, provides the setting for our proposal.

1. Introduction

IEEE 830-1998 [11] recommendations are the milestone concerning the contents which are mandatory in a Software Requirements Specification (SRS). This standard suggests several possible organizations for specific software requirements. Nevertheless, when the amount of requirements and/or the complexity are considerable, those suggestions are not enough. Requirements organization and presentation are crucial to facilitate their maintenance and ensure other desirable features such as: correction, completeness, consistency and traceability. Along the requirements elicitation and specification process, several stakeholders are involved (both from the customer and technical side) every one with their own interests and views of the system, which ought to be rightly represented and reconciled in the SRS.

A requirement is a capacity that software should exhibit or a condition that should be met. This capacity or condition can be expressed at different abstraction or detail levels. Concretely, we distinguish two levels: *goals* and *requirements*. Goals allow one to establish, by means of refinement and composition processes, a derivation graph from software interests or goals until specific requirements. On the other hand, *requirements* are detailed enough as to be assigned to a software component and lately be verified. *Goal Oriented Requirements Engineering* [13] employs this strategy to perform this refinement from goals until requirements and their subsequent operationalization in software elements.

In the literature, different paradigms cope with the *separation of concerns* of a system in order to provide support for evolution adaptability, comprehensibility, etc. These *concerns* can range from non-functional features, as security or fault tolerance, to functional features. Aspect Oriented Software Development (AOSD) [1] is one of these approaches. AOSD provides a set of techniques that allow managing those interests that appear scattered along the system and crosscut several elements. AOSD identifies these *concerns* realizing them as *aspects* and managing them explicitly.

In the AOSD context, several proposals have been introduced at different abstraction levels, both during implementation [12] and design [21] that define an aspect as an additional constructor of the language. It is associated to the constructor "class" to manage the crosscutting that can appear in the methods specification for a class. Several works have also been proposed at the requirements level as Aspect Oriented Requirements Engineering (AORE) [20]. In this proposal, an aspect is a requirement which is related to a set of other requirements, but that is separately specified. By means of

[*] This work has been funded by the Spanish CICYT project DYNAMICA TIC2003-07776-C02-02

a technique known as *weaving,* it manages this crosscutting.

The aim of this work is to present our proposal to organize software requirements, by integrating Aspect Oriented techniques within a Goal Oriented approach for requirements, as we previously sketched in [17]. Furthermore, the ISO/IEC 9126 [10] standard is used as a starting point to define the concerns. ATRIUM [16], a methodology to concurrently define requirements and software architecture, provides the setting for our proposal.

The reminder of this work is organized as follow. Section 2 briefly describes the ATRIUM methodology. Section 3 introduces the ISO/IEC 9126 standard and how it is used in our proposal. Section 4 presents in detail the set of tasks that drive the goals model definition. The way we apply the set of tasks to a case study is shown in section 5. Some related works are described in section 6. Eventually, the achieved conclusions and future works round up the work.

2. ATRIUM: Requirements and Software Architectures

ATRIUM is a methodology oriented to the concurrent definition of Software Architectures (SA) and Requirements. In ATRIUM, decisions at architectural level are made to satisfy specific software requirements. With this aim, ATRIUM provides the analyst with guidance, along an iterative process, from an initial set of user/system needs until the instantiation of the architecture, specified by means of a PRISMA model [18]. PRISMA is an architecture description language that allows us to define dynamic architectures.

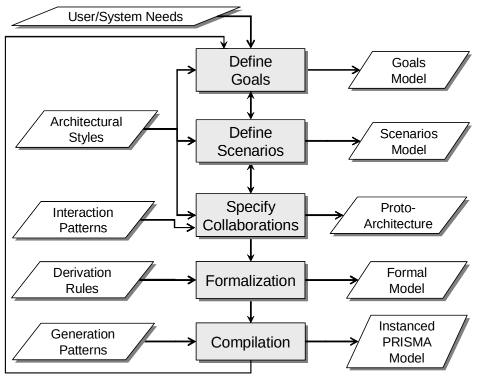


Figure 1 ATRIUM: Activities and Artifacts

ATRIUM entails five activities (Figure 1) to be iterated on in order to define and refine the different artifacts and allow the analyst to reason about partial views, both of requirements and of architecture. The *Define Goal* activity (see section 4) allows identifying the different concerns of the software and the crosscutting between them. These concerns are candidate to be classified as aspects, in the PRISMA specification, and realize them through aspects integrated into components and/or connectors.

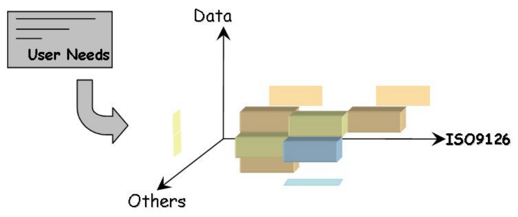


Figure 0 Unfolding a Software Specification

3. ISO/IEC 9126: Selecting and Identifying Concerns

Quality criteria, used for the software products assessment, are highly related to the requirements specified in their SRS. This means a practical binding with the global organization of the SRS in order to facilitate the subsequent evaluation of the software quality. In this sense, the ISO/IEC 9126 standard is an important reference as software quality model by defining a set of features that can be required of quality software. This reason makes the ISO/IEC 9126 especially suitable as taxonomy for concerns. It provides an initial framework to elicit and organize goals and requirements. In this way, as the informal software needs are elicited, they can be analyzed, broken down and organized. This allows managing the specification crosscutting, by reducing or removing other drawbacks such as: redundancy, inconsistencies, etc. At the same time, weaving relationships can be defined in order to re-establish a tangled representation whenever it is needed.

ISO/IEC 9126 determines three software quality aspects: process quality, product quality and product in use quality. The main aim is just the latter, i.e., to notice the product quality through the effect that its use causes. The quality in use depends on or is influenced by the internal and external characteristics of the software product. These characteristics are affected by the software construction process. With regard to the specific requirements, described in the SRS, we are interested in the characteristics defined for the software product quality and for the quality in use. These characteristics are listed in Table 1.

We have to notice that from the software requirements perspective this taxonomy goes beyond the traditional classification of functional and non-functional requirements which is not a meaningful contribution to the requirements organization. In fact, most of the typical functional requirements can be set with the *suitability* sub-characteristic. On the other hand, the software product capacity to satisfy the standards, conventions or regulations are broken down as sub-characteristics of type "compliance" below each quality software product sub-characteristic. Although the ISO/IEC 9126 provides us with a wide set of concerns, this set can be extended if needed. In this case, several alternatives arise when we apply our proposal:

a) Considering a new dimension for organizing goals/requirements, additional to the taxonomy proposed by ISO/IEC 9126. This option could be of interest whenever the additional characteristics are as relevant as those already considered and whether the crosscutting with them could be high.
b) Including a new characteristic/sub-characteristic for extending the taxonomy. This alternative would be recommended when the aspects which are dealt with are not as relevant as those considered and/or it is not expected that the crosscutting could be so high.
c) Dealing with this element as an attribute of the goal/requirement. This option is suggested when it is coped with aspects which are not so relevant (they are neither exactly goals nor requirements) or do not involve an important crosscutting. However, they are particularly interesting to group and present goals/requirements.

Table 1 Quality Characteristics of the ISO/IEC 9126

Quality Type	Characteristic	Sub- Characteristic
Software Product Quality	functionality	suitability
		accuracy
		interoperability
		security
		compliance
	reliability	maturity
		fault tolerance
		recoverability
		compliance
	usability	understandability
		learneability
		operability
		attractiveness
		compliance
	efficiency	time behaviour
		resourse utilisation
		compliance
	maintainability	analysability
		changeability
		stability
		testeability
		compliance
	portability	adaptability
		installability
		co-existence
		replaceability
		compliance
Quality in use	effectiveness	
	productivity	
	safety	
	satisfaction	

The IEEE 830-1998 offers several criteria and guidelines to organize specific requirements. It recognizes that there is no an optimum organization to be applicable to every system. Between the mentioned organization criteria are: operation system mode, user type, problem entities, system services, stimulus, answer and/or functions hierarchies. Therefore, as it is recommended by the *c)* alternative, those elements can be dealt as attributes of goals/requirements instead of extending the taxonomy. In this way, it is possible to offer a view related to the joins based on these elements, although they are not elements of the initial taxonomy.

For instance, we could deal with the section "logic requirement of the database" (included in the IEEE 830-1998) as an aditional dimension (called *data*). Figure 2 illustrates the framework extension with new dimensions.

4. Goals Model and *Concerns* of the system

Both identification and specification of the different *concerns*, and their possible *crosscutting*, are addressed, in this work, with the definition of the tasks described in Figure 3, which integrate the *activity 1*. The output of this activity is the Goals Model, which was previously described in [15]. Its conception was influenced by the NFR Framework [3] and the KAOS proposal [4], although it integrates both functional and non-functional requirements. Furthermore, it plays an important role for the aspect identification as we previously stated in [17]. The Goals Model allows one to specify not only *goals*, *requirements* and *operationalizations* but also the *crosscutting* which can appear. For the Goals Model construction, the ISO/IEC 9126 quality model, described above, is used as an instantiable framework, providing the analyst with an initial description of the set of *concerns* of the system.

4.1. Defining the Goals Model

Figure 3 establishes the workflow for the Goals Model elaboration along with the input artifacts needed for its realization. Although Figure 3 shows only a sequential flow to apply the tasks, in practice, its application is iterative. Involved tasks, in the Goals Model, are run with every new identified or selected goal.

The first task deals with the *Identification/Selection of Goals*. As we can observe in Figure 3 the standard ISO/IEC 9126 is an input for this task. This model provides an initial view of the *concerns* that could be meaningful for the system. In this way, the analyst can iteratively select what he/she considers proper and initiate its specification and refinement. As we elicit the requirements, it could be convenient to incorporate new *concerns* to properly include additional goals/requirements.

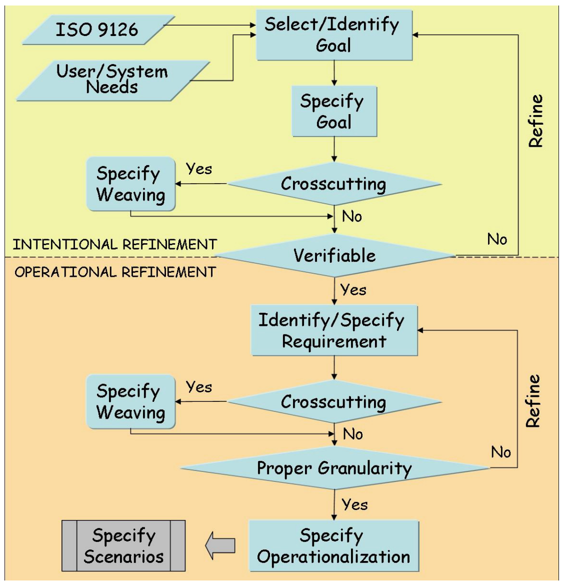


Figure 3 Workflow to specify goals and requirements

User/System needs are other input for this task. They provide us with information to identify new goals/requirements. Every goal, established by means of this process, is aligned with the specification of *concerns* determined by the ISO/IEC 9126 model, and acts as a node for the graph definition. This will provide us with a twofold advantage; on one hand to facilitate the understanding of the specification, and, on the other hand, to drive the elicitation and analysis process.

In the task *Specify Goal*, the attributes, that constitute it, are not only established, such as its name, priority, etc., but also composition relationships (AND/OR/XOR) among goals ([15] offers a full description of the goals model).

When crosscutting of goals/requirements is identified, task *Specify Weaving* is performed in order to establish weaving relationships. These relationships can be stereotyped according to the traditional AOSD weaving mechanisms, i.e., *before* and *after*. This allows us to express how a piece of goal/requirement specification (from the aspect point of view) is incorporated inside some goal/requirement specification. Other more specific weaving relationships could be used (like in [19]), but we suggest to do this refinement in the specific domain context of the system.

This refinement process of goals goes on until the goal is assignable to a system agent. At this moment, we change from an intentional refinement to an operational refinement, and consequently to the specification of a requirement. Another difference between a goal and a requirement stems from the latter ones have to be verifiable. In this way, *Identify/Specify Requirements* follows a similar process to that observed for goals, through the definition of attributes and composition relationships. Additionally, weaving relationships can also be established.

After the requirement specification, the next task is *Specify Operationalization*, that is, the definition of the agent or set of agents that collaborate in its realization. During the Goals Model construction, the operationalization is only a description of the proposed solution for the realization of a requirement, working this description as an input for the ATRIUM activity *Define Scenarios*. The latter cope with the whole definition of the solution through the relevant scenario specification. It is introduced in the Goals Model in order to allow us to describe the relationships between that solution and the already defined requirements in the Model. In this way, we can denote how a solution can contribute to positively realize a requirement and negatively to others. Thanks to these relationships, we achieve a more exhaustive analysis of the set of possible solutions.

5. Case Study

This section illustrates how we have applied our proposal in the context of a real system. This work was developed thanks to the collaboration with a group involved in the European Project Environmental Friendly and cost-effective Technology for Coating Removal (EFTCoR) [5]. The main scenario of this project is the hull maintenance operations. Mainly, it addresses operations of coating removal, washing and re-painting of hull of ships by using a family of robots, that either performs different operations or the same operation but in a different way.

These maintenance operations have a high impact both economical and environmental. The former is related to the time that the ship must go into the dry dock and to the costs derived of its maintenance. The later is due to the generated residues along the operations. Furthermore, these processes are very hazardous for operators.

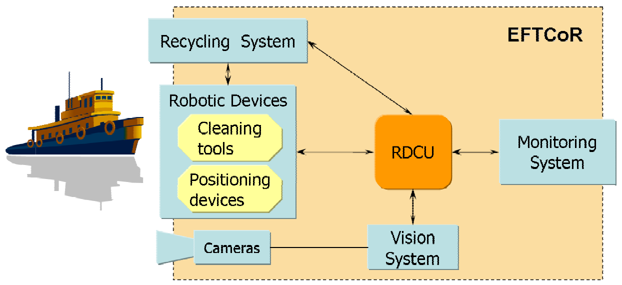
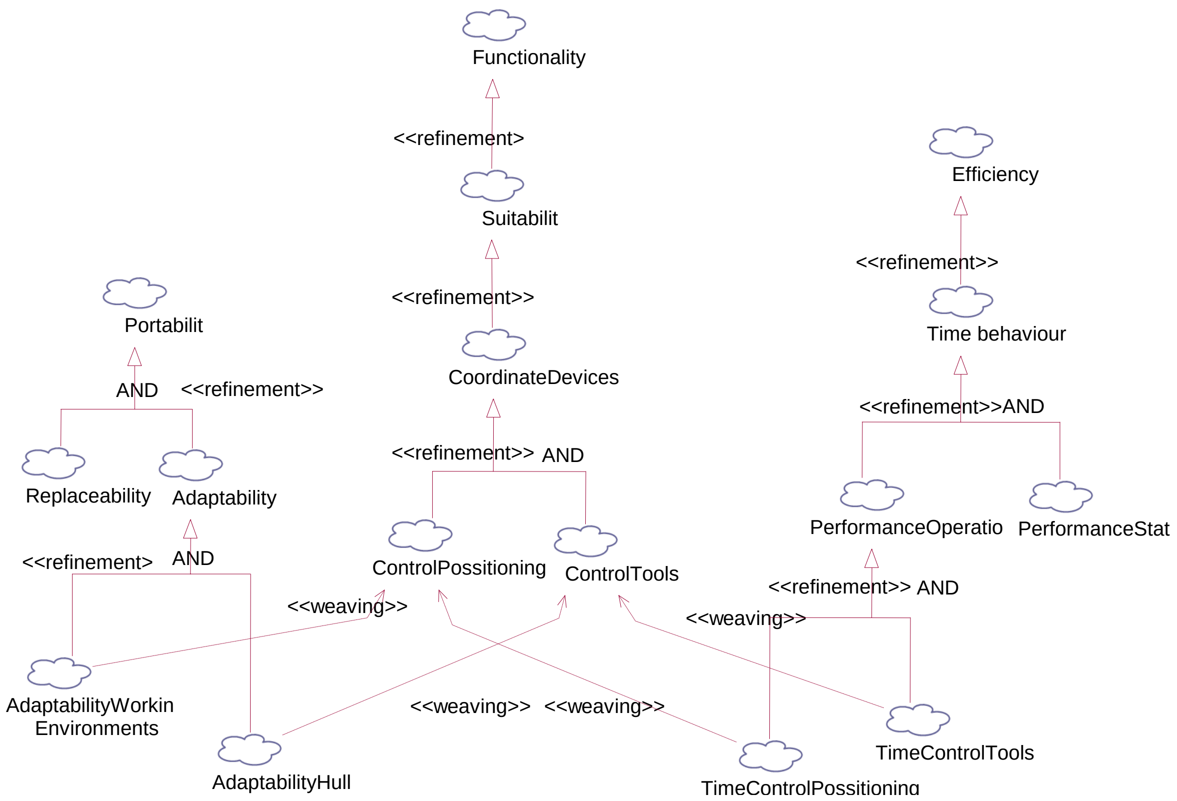


Figure 4 Teleoperation Robotic System for Hull Maintenance Operations



The identified robotic teleoperation platform [5] is integrated by the next subsystems (illustrated in Figure 4):

a) *Monitoring System*: encompasses the functionality concerning to the informational and managerial needs related to ship maintenance operation that is going to be accomplished.
b) *Vision System*: allows the hull inspection of the working areas and provides information for automatically moving the robotic devices along the hull.
c) *Recycling System*: retrieves the residues from the working areas and recycles them.
d) *Robotic Devices Control Unit*: interacts with the others robotic devices with the aim of getting the needed information to control the different devices (positioning systems and cleaning tools) to be used in the maintenance tasks. It is accomplished according to the commands introduced by the operator.

Our case study focuses on the Robotic Devices Control Unit (RDCU, Figure 4). Its architectural definition is highly relevant because the fact that several constraints have to be satisfied in order to allow a dynamic behavior of the system. This dynamism allows the EFTCoR to replace, at run time, each cleaning tools and positioning devices. Either change or operation has to be secure, providing a means to stop it if any damage can be produced to the equipment, the environment or the operator. Moreover, every operation has to be scheduled to accomplish hard deadlines.

5.1. Specifying the Goals Model

Next, some sentences about the established needs for RDCU, extracted from [5], are presented:

Positioning systems and tool can work simultaneously. The RDCU is responsible of co-ordinating their actions according to their operational states, the mission parameters, and the current state of the environment.
(1) The operational commands for positioning systems and activating the tool can be easily and efficiently issued to the system. *(2)The system should react to such commands efficiently. In some cases, the execution time of the commands should be smaller than a given deadline.*
(1)The possibility of using different coating removal technologies (blasting, water pressure). Though the chosen technology for the EFTCoR project is blasting, the RDCU should be open to incorporate cleaning tools based on other technologies. *(2) The possibility of using the system for different maintenance tasks, including at least the fresh water washing before blasting and the painting after blasting.* *(3) The possibility of using different positioning systems and different combinations of primary and secondary positioning systems.* *(4) The possibility of using different tools for the same or different processes (already considered in the first point).*

From the previous statements we can observe how the crosscutting appears in the specification, as for instance the goals related to *Efficiency* and *Adaptability*. Both goals are applied to other goals as *ControlPositioning* or *ControlTools*. When the task *Identification/Selection of Goals* was applied by using the previous sentences as input, it generated as a result the next goals list that appears on Table 2 (more details about the applicable attributes can be found in [15])

The graph on Figure 4 shows (part of) the Goals Model where the refinement of goals from Table 2 is reviewed. In this way, we observe how *Portability*, *Functionality* and *Efficiency* are some of the selected characteristics to become concerns for the EFTCoR system. Furthermore, this figure shows us some of the relationships of refinement that were established. For instance, the AND relationship for the goals *AdaptabilityWorking-Environment* and *AdaptabilityHullMaintenaceOperation* that was introduced to satisfy *Adaptability*. On the other hand, a weaving relationship has been established between *AdaptabilityWorkingEnvironment* and *Control-Possitioning.*

Table 2 (Partial) Description of Goals for EFTCoR

GOAL	DESCRIPTION
Functionality	The system has to provide functions which meet stated and implied needs
Suitability	The system has to provide an appropriate set of functions for specified tasks and user objectives
CoordinateDevices	The RDCU has to coordinate robotic devices according to the current mission procedure and global system state
ControlPossitioning	The RDCU has to control the positioning devices
ControlTools	The RDCU has to control the tools attached to positioning devices (blasting head, painting tool, etc)
Efficiency	The system should use efficiently their resources
TimeBehaviour	The system should respond with appropriate speed
PerformanceOperation	The system should respond with appropriate speed to the operation commands
Portability	The system should be able to be transferred from one environment to another
Replaceability	The system should be able to use different software product for the same purpose
Adaptability	The systems has to be adaptable for different specified environments
AdaptabilityWorkingEnvironments	The system operation has to be adaptable to different working environments
AdaptabilityHullMaintenanceOperation	The system operation has to be adaptable to different hull maintenance operations

6. Related works

There are many works related to the management of aspects at different abstraction levels, at implementation as well as design level. At the requirements level, several proposals have focused on how to identify and specify concerns of the system. They also focus on how to determine which concerns will be realized as aspect, in other derived artifacts, in such a way that the closure property [6] is satisfied.

Grundy [9] has defined a proposal called Aspect-Oriented Component Engineering (AOCE), in order to define and develop software components from requirements until design, implementation and deployment. In this proposal, components are integrated by aspects, but there is no explicit specification about how these aspects are a realization of the aspects at the requirements level.

Rashid et al [20] have proposed a model for requirements engineering that identifies the concerns at an early stage of development, along with the requirements. This model allows relating and establishing the impact of these early aspects on later stages of the development. Nevertheless, this proposal does not allow recreate the specification to offer a better comprehension for the validation end user.

Brito et al [2] proposal is the closest one to our approach. It introduces a goals model, the NFR Framework [3] concretely, to specify non-functional requirements along with another technique, such as use cases or viewpoints, to specify functional requirements. In this way it extends the framework with a new type of relationship called *required concerns* in order to determine a potential crosscutting with other requirements. However, our proposal provides the analyst with a unique artefact to specify both kinds of requirements. It facilitates both its use and comprehensibility.

On the other hand, no stated proposals provide the analyst with an initial framework which helps him/her to identify the concerns of the system. Finally, another significant difference is related to the uniform management of goals/requirements, with regard to the applicable categories for its classification. Nevertheless, other proposals explicitly focus on the distinction between functional and non-functional, by employing different techniques to describe both kinds of goals/requirements. This entails a high cost in terms of legibility and maintainability.

Additionally, most of the Aspect Oriented requirements proposals, such as [19, 14], apply separation of concerns techniques after identifying the crosscutting in the specification. In this way, they offer a solution to improve the original specification. In our proposal, thanks to the use of a Goal Oriented approach, that allows us to begin the software requirement specification at a level of concerns, we can deal with the goals and requirements relationships detecting and managing the crosscutting among them by using weaving relationships.

7. Conclusions and future works

In this work, we have presented a proposal to elicit and analyse requirements that integrates Goal Oriented and Aspects Oriented approaches. Owing to the Goal Oriented approach, an explicit traceability is preserved from goals towards requirements, and from the latter to the operationalizations that realize the software components. On the other hand, the Aspect Oriented approach allows a smart and effective management of the crosscutting that can appear in the SRS when goals and requirements are tangled. By defining weaving relationships between goals/requirements elements an optimal representation is achieved, that can allow to recreate the original representation whether it is needed. A workflow has been defined that integrates both approaches, detailing a set of tasks. It provides a guide for the elaboration and organization of the requirements.

Another advantage that offers our proposal is the use of the ISO/IEC 9126 as a starting point to establish the possible concerns. Additionally, it is possible to tailor, in terms of content, the SRS to the IEEE 830-1998 but with meaningful advantages for elaboration and organization of the requirement specification.

Several challenges are to be faced. One of them is related to the development of a tool that provides our proposal with a proper support. The flexibility for the requirements representation, regarding several criteria, is one of main factor in this development. With this aim, we are exploiting the structure of characteristics as well as using attributes which are defined for goals and requirements.

Another point of interest, for future works, is related to the weaving relationships. Concretely, we are concerned with its suitability to the set of offered relationships along with their semantic. According to this topic, the emerging issues from the case study will facilitate its verification and validation. Specially when, following the ATRIUM stated process, the established weaving relationships and the identified concerns and requirements will have traceability to aspects in architectural artefacts.

References

[1] Aspect-Oriented Software Development, http://www.aosd.net

[2] I. Brito, A. Moreira, "Integrating the NFR framework in a RE model", Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, March 22, 2004 - Lancaster, UK in conjunction with 3rd Aspect-Oriented Software Development Conference.

[3] L. Chung, B. A. Nixon, E. Yu and J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic Publishing, 2000.

[4] A. Dardenne, A. van Lamsweerde, and S. Fickas: "Goal-directed Requirements Acquisition". Science of Computer Programming, 20:3-50, 1993.

[5] EFTCOR: Environmental Friendly and cost-effective Technology for Coating Removal. European Project within the 5th Framework Program (GROWTH G3RD-CT-00794), 2003.

[6] T. Elrad, M. Aksits, G. Kiczales, K. Lieberherr, H. Ossher, "Discussing aspects of AOP", Communications of the ACM, Vol. 44 , Issue 10, pp. 29 – 32, October, 2001.

[7] T. Elrad, R. E. Filman and A. Bader, "Aspect-oriented programming: Introduction", Communications of the ACM, Vol. 44 , Issue 10, pp. 29 – 32, October, 2001.

[8] C. Fernández, J.A. Pastor, P. Sánchez, B. Álvarez, A. Iborra: "Co-operative Robots for Hull Blasting in European Shiprepair Industry. Robotics and Automation Magazine (RAM)", special issue on Industrial Robotics Applications & Industry-Academia Cooperation in Europe. New Trends and Perspectives. (To appear in September 2004)

[9] J. Grundy, "Aspect-Oriented Requirements Engineering for Component-Based Software Systems", In Proceedings of the IEEE International Symposium on Requirements Engineering, June 07 - 11, 1999, Limerick, Ireland.

[10] ISO/IEC Standard 9126-1 Software Engineering- Product Quality-Part1: Quality Model, ISO Copyright Office, Geneva, June 2001

[11] IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications, In Volume 4: Resource and Technique Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection.

[12] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. M. Loingtier and J. Irwin, "Aspect-Oriented Programming". In the Proceedings of the European Conference on Object-Oriented Programming, Finland. Springer-Verlag LNCS 1241. pp. 220-242, June 1997.

[13] A. van Lamsweerde "Goal-Oriented Requirements Engineering: A Guided Tour", Invited Paper for 5th IEEE International Symposium on Requirements Engineering, Toronto, August, 2001, pp. 249-263.

[14] A. M. D. Moreira, J Araújo, I. Sofia Brito, "Crosscutting quality attributes for requirements engineering". In Proceedings of the 14th international conference on Software Engineering and Knowledge Engineering, pp. 167-174, July 15-19, 2002, Ischia, Italy.

[15] E. Navarro, I. Ramos and J. Pérez: "Goals Model-Driving Software Architecture", 2nd International Conference on Software Engineering Research, Management and Applications, May 5-8, 2004, Los Angeles, CA, USA.

[16] E. Navarro, I. Ramos and J. Pérez: "Software Requirements for Architectured Systems". Proc. 11th IEEE International Conference on Requirements Engineering, pp. 365-366, September 8-12, 2003, Monterey, CA, USA.

[17] E. Navarro and I. Ramos, "Requirements and Architecture: a marriage for Quality Assurance", VIII Jornadas de Ingeniería del Software y Bases de Datos, Alicante, 12-14 Noviembre 2003

[18] J. Pérez, I. Ramos, J. Jaén, P. Letelier and E. Navarro, "PRISMA: Towards Quality, Aspect Oriented and Dynamic Software Architectures", In Proceedings of the 3rd IEEE International Conference on Quality Software, Dallas, Texas, USA, pp. 59-66, November 6 - 7, 2003.

[19] A. Rashid, A. Moreira, J. Araújo, "Modularisation and composition of aspectual requirements", In the Proceedings of the 2nd international conference on Aspect-oriented software development, pp. 11-20, March 17 - 21, 2003, Boston, Massachusetts.

[20] A. Rashid, P. Sawer, A. Moreira and J. Araújo, "Early Aspects: a Model for Aspect-Oriented Requirements Engineering", In the Proceedings of the International Conference on Requirements Engineering, pp. 199-202, September 9-13, 2002, Essen, Germany.

[21] J. Suzuki and Y. Yamamoto, "Extending UML with Aspects: Aspect Support in the Design Phase", AOP Workshop at ECOOP'99, Lisbon, Portugal, 1999.