SURVEY ON EARLY ASPECTS APPROACHES: NON-FUNCTIONAL CROSSCUTTING CONCERNS INTEGRATION IN SOFTWARE SYSTEMS

DENISE LAZZERI GASTALDO BOMBONATTI, SELMA SHIN SHIMIZU MELNIKOFF
Escola Politécnica da Universidade de São Paulo – USP
Av. Professor Luciano Gualberto, travessa 3, n 380.
Sao Paulo - Brazil
{denise.gastaldo, selma.melnikoff}@poli.usp.br

Abstract - Frequently non-functional requirements crosscut software systems and can deeply affect software project decisions. Aspect-oriented approaches aim to identify and specify crosscutting concerns in separate units, named aspects. In this context, this paper proposes to compare early aspect-oriented approaches that integrate non-functional crosscutting concerns in software systems.

Key Words: - Non-functional requirement, aspect-oriented, aspect, concern, crosscutting concern.

1. Introduction

Nowadays, software development and application domains complexity have grown incrementally. In this context, non-functional requirements have been increasingly accepted as crucial to software systems success [9].

Aspect-oriented approaches aim to reduce software development, maintenance and evolution costs [1], and improve software systems modularity, comprehensibility, maintainability and reusability [16]. Besides those benefits, aspect-oriented approaches reduce tangling and scattering phenomena by crosscutting concerns identification, separation and composition.

A concern is an interest, which pertains to the system development, its operation or any other matters that are critical or otherwise important to one or more stakeholders [2]. Concerns identification improves change control, maintainability, reduces complexity, helps to define responsibilities and keeps the software traceability [1] [3]. Concerns separation allows important points in the system to be handled separately [4]. Concerns composition ensures complete and coherent integration [2].

In order to reduce tangling and scattering, aspect-oriented programming was created to improve the separation of crosscutting concerns [5]. However, the decision to separate concerns at the implementation phase is late and costly, because many project decisions have been already taken. In this context, many aspect-oriented software development methods were proposed to identify, separate and compose crosscutting concerns from the beginning of lifecycle [4]. Additionally, many early aspect detection methods were proposed to be applied at early stages, at requirements level [6]. At early stages, candidate aspects can be identified. The term candidate aspect has been used to represent an early aspect that may or may not be modeled as an aspect in a later design of the system [7].

The proposal here is to compare early aspects approaches that integrate non-functional concerns in software systems. A comparison criterion is defined in section 2. A brief overview of the approaches is presented in section 3. The evaluation results are presented in section 4. Finally, section 5 will conclude the paper.

2. Comparison Criterion

The comparison criteria were established based on [8], which defines a set of activities of an integrated AORE (Aspect-Oriented Requirements Engineering) approach:

• Concern elicitation: the requirements engineer defines a set of issues of interest to the stakeholder. This can be done through discussions with stakeholders, interviews, etc.
• Concern identification: the crosscutting concerns are identified and separated from software system concerns.
• Concern representation: the concerns are represented using a graphical representation, lists or templates.
• Concern composition: concerns are composed and the composition result is used as a basis to identify
conflicts. The conflicts between concerns are solved through a trade-off analysis.

The approaches will be compared in the way these activities are performed.

3. Early Aspect-Oriented Approaches Overview

Early aspect-oriented approaches which integrate non-functional concerns in software systems were selected to be compared in this survey. In the most part of the approaches, use cases [14] and NFR Framework [15] are considered as main support techniques. In this section, an overview of each approach will be presented.

3.1 AORE with UML

AORE with UML [10] is a UML-based proposal based on the general aspect-oriented requirements engineering process presented in [1]. This general approach introduced the term AORE and the early aspects concept. The focus of AORE with UML is on non-functional crosscutting concerns separation and composition.

Concern elicitation is performed through meetings and discussions with stakeholders. Non-functional concerns are identified as a kind of restriction that the system must satisfy. Non-functional crosscutting concerns are identified as a non-functional requirement that transverses or affect more than one use case. These crosscutting concerns are described in specification templates. New use cases with non-functional crosscutting concerns are composed in the use case model with new relationship stereotypes, such as <<wrappedBy>>, <<overlapedBy>> and <<overridedBy>>. Conflicts are detected through a negative or positive contribution analysis between concerns, and the decision is based on negotiation among stakeholders. Further aspect impacts on specification, architecture, design - named aspects dimensions - are identified.

3.2 Aspectual Use Case Driven Approach

Aspectual use case driven approach [11][12][13] is similar to AORE with UML approach. The difference is in functional and non-functional crosscutting concerns separation and composition.

Concern elicitation is performed through requirements inspection and information obtained from system stakeholders. Non-functional concerns are identified according to stakeholders needs, and functional concerns are identified through use case technique [14], using <<includex>> and <<extendx>> relationships between use cases to promote the externalization of requirements that are scattered. Crosscutting concerns are detected if they constrain, extend or are included by more than one use case. Non-functional concerns are described in specification templates and functional concerns are described in use case model. Non-functional crosscutting concerns are composed in the use model with a new use case relationship stereotype named <<constrain>>. The composition is also proposed in sequence diagrams with after and before operators. There is no clear trade-off analysis proposed in this approach.

3.3 NFR Framework integrated in Requirements Engineering Model

NFR Framework integrated in requirements engineering model [3] was proposed based on aspectual use case driven approach, differing in composition activity.

Concern elicitation is performed through a complete and detailed analysis of the documentation and stakeholder information. Functional and non-functional concerns are identified using the most appropriate approach, such as use cases [14], viewpoints, NFR Framework [15]. If the concern is requested by more than one concern, it is considered a crosscutting concern. The concerns are represented by specification templates, use case model and SIG (Softgoal Interdependency Graph) [15]. In these templates, the type of contribution, positive or negative, and priorities are identified. In the concern composition, match points are identified in a matrix (stakeholder x concern). If in the same cell of this matrix, more than one crosscutting concern is detected, a conflict and priority management will be requested to be negotiated with stakeholders. A set of composition operators are applied in identified match points.

3.4 AORE integrated with NFR Framework

AORE integrated with NFR Framework [16] [17] was initially focused on non-functional crosscutting concerns separation and composition. An adaptation was proposed in [18] to deal with functional and non-functional crosscutting concerns.

Concern elicitation is performed through system information analysis, stakeholder needs, regulations, domain information and business rules. NFR softgoals [15] are considered non-functional crosscutting concerns by nature, and functional crosscutting concerns are identified through use case technique [14]. Functional and non-functional crosscutting concerns are considered aspectual requirements. Non-functional crosscutting concerns are represented in SIG [15] and functional ones are represented in use case model. Composition operators are defined, such as overlap, override and wrap to be part of composition tables, which define the points where aspectual requirement behaviour will be added. Trade-off analysis is performed to identify conflicts between crosscutting concerns. Project restrictions and their impacts on architecture or implementation are not considered in this approach.
3.5 Aspect-Oriented Software Development with Use Cases

Aspect-oriented software development with use cases [4] proposes that a use case is a crosscutting concern by nature, since the realization of each use case affects several classes. It also defines the concepts of use case slice that contains the details of a use case, the concept of use case module, which contains all the use case information across the development phases.

Concern elicitation is performed by use case technique [14]. Non-functional crosscutting concern is represented by a special type of use case, named infrastructure use case. In this approach, there are two types of separation of concerns, by peer use cases, that are independent use cases with no reference to other use cases, and extension use cases. The composition is performed in extension use cases, by the use of before, after and around operators, and during compilation time or runtime. There are guidelines to deal with overlaps with a unit named non-specific use case slice, which will group common behaviour of several use cases.

3.6 Concern-Oriented Requirements Engineering Model

Concern-oriented requirements engineering (CORE) [19] [20] was proposed based on AORE.

Concerns elicitation is based on the stakeholder view about the system functionality. Concerns can be identified using techniques such as use cases [14], viewpoints and goals [15]. Concerns that affect several other concerns are classified as crosscutting concerns. The specification is represented by XML language with some pre-defined tags. Concerns projections are identified by a matrix (concern x concern), and further composition is defined based on these projections. The composition is also represented in XML schema. Contributions between concerns are analysed as positive or negative, and weights are attributed to conflicting concerns in order to define priorities with stakeholders. Concerns dimensions are defined in order to determine their influence in later development stages.

3.7 Aspect-Oriented Development Model with Traceability Mechanism

Aspect-oriented development model with traceability mechanisms [21] aims to present an approach that complements separation and composition of crosscutting concerns with traceability mechanisms for dynamic and static views. Dynamic view traces uses cases and scenarios, and static view traces conceptual classes.

Concerns elicitation is performed through discussion with stakeholders, proposals review, prototypes and requirements elicitation meetings. Functional concerns are identified by use case technique. Non-functional concerns are identified by discussions with stakeholders. If a use case is included by several use cases, then it will be a crosscutting concern. Non-functional concerns are classified as crosscutting concerns if it is considered as a global system property. Concerns are represented by specific traceability models that connect requirements, use cases, scenarios and classes. The composition is performed by adding a new use case with a crosscutting behaviour identified by <<CCR>> stereotype. Conflicts are managed by assigning priorities, and guided by expert opinion.

3.8 Linking Goals to Aspects

Linking goals to aspects [22] proposes that satisficing OR decomposed goals in the KAOS model usually leads to tangled representations. These goals should be implemented in the aspect-oriented manner. KAOS (knowledge acquisition in automated specification framework) consists in a set of entities representing requirements as goals to be achieved.

Concerns elicitation and identification is based on code reengineering [22]. This approach is based on some steps. The first one applies a technique of mining aspects to identify crosscutting features. The second step deals with a consolidation of application requirements in KAOS concepts. The third step performs a comparison between the goal decomposition and actual code decomposition in order to identify patterns and give rise to the use of aspects. The last step consists of a validation of these patterns in a completely different type of application. In this work patterns and aspects identification starts from code level followed by a KAOS representation. Trade-off analysis is proposed to deal with conflicts, but no clear guidelines are presented.

3.9 Identifying Candidate Aspects with I-star Approach

I-star framework is a technique that intends to capture relevant organizational environment information to the system under development. Identifying candidate aspects with I-star approach proposes to identify candidate aspects in early stages using I-star models [23]. An integration of I-star models and UML is proposed. Is it advocated that UML is not adequate to deal with all the important matters required to requirement capture phase.

Concerns elicitation and identification is performed by the use of I-star approach and use case approach. Crosscutting concerns in use case approach can be identified when a use case is included by more than one use case, or if there is a use case that extends more than one use case. Crosscutting concerns in I-star approach are identified if a model element is required by, or affects several other model elements modifying their original behaviour.

A use case model is derived from adapted I-star approach and a set of crosscutting concerns is identified.
This approach intends to handle with softgoals influences and presents a proposal to this objective. Usually softgoals deals with non-functional requirements, and according to many authors [1] [16] [17] [19] [20] non-functional requirements are potentially candidate aspects. Composition activity is not forecasted in this approach.

4. Evaluation

The non-functional crosscutting concerns identification of AORE with UML approach (section 3.1) is not supported by a clear guideline. The trade-off analysis is also subjective because conflict decision may depend on the stakeholder experience. No heuristics is defined to determine how to establish aspect dimensions.

The identification of non-functional concerns in aspectual use case driven approach (section 3.2) is not supported by a clear guideline, either. No trade-off analysis is proposed to determine conflicts between crosscutting concerns. In NFR Framework integrated with requirements engineering model (section 3.3) the trade-off analysis is executed, but non-functional concerns identification is still vague and abstract.

AORE integrated with NFR Framework (section 3.4) deals with functional and non-functional crosscutting concerns, but no heuristics is defined to deal with aspect dimensions impacting other development phases, such as architecture or implementation.

Aspect-oriented software development with use cases (item 3.5) is not focused on non-functional concerns. The decisions related to infrastructure use cases are taken in later development phases. No guideline is available to evaluate the impact of an infrastructure use case in the use case model. Overlap analysis is performed, but may not be sufficient to evaluate impacts between crosscutting concerns. This approach requires detailed design information, not available at the requirements phase, to confirm that a use case slice crosscuts many classes.

Concern-oriented requirements engineering (section 3.6) deals with functional and non-functional concerns. The use of XML should require a tool to facilitate this approach application.

Aspect-oriented development model with traceability mechanism (section 3.7) deals with functional and non-functional concerns, but traceability representation will need a tool support to facilitate dynamic and static representation.

Linking goals to aspects (section 3.8) deals with functional and non-functional concerns, code reengineering, identification and representation of crosscutting concerns. This approach is recommended when the code is available. For new developments other approaches should be more appropriated. Trade-off analysis is proposed, but composition guidelines are not available.

Identifying candidate aspects with I-star approach (section 3.9) also deals with functional and non-functional concerns, identification and representation of crosscutting concerns; however, composition activity is not forecasted.

An additional analysis was performed to verify AORE activities execution by the approaches presented. Table 1 summarizes this information. Some approaches do not sufficiently perform composition activity with trade-off analysis. The AORE with UML is the only approach which does not deal with functional crosscutting concerns.

The comparison of the approaches also allowed the analysis of each AORE activity:

- **Concern elicitation**: all approaches apply elicitation activities, such as meetings, documentation review, stakeholder needs identification, reengineering, etc.
- **Concern identification**: all approaches identify concerns, but some of them handle crosscutting concerns in a high-level, vague, abstract and informal manner. A clear understanding of a concern meaning and the way it can be satisfied in the software system may not be achieved. Some of the approaches consider non-functional concerns as crosscutting by nature. This must be carefully evaluated case by case.
- **Concern representation**: all approaches represent and specify concerns, but some of them require an additional tool support to facilitate the work.
- **Concern composition**: almost all approaches consider concern composition activities. The trade-off analysis is executed by most approaches, but in some cases, it depends on stakeholder negotiation and experience.

The approaches that apply NFR Framework [15] present a clear benefit in non-functional concern handling. Non-functional concerns can be considered objectively, with a method and tool support.

No approach presented any solution to deal with aspect dimensions, such as architecture and implementation. No guidelines were established to operationalize these impacts in further development phases.
5. Conclusion

This survey discussed many approaches to deal with non-functional concerns integration in software systems. Many opportunities can be addressed in early aspect-oriented area and many researches can be developed to fill detected gaps.

Acknowledgements

This work is supported by Escola Politécnica da Universidade de São Paulo.

References


