INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 2 JULY 2015

GOALS TO ASPECTS: DISCOVERING ASPECTS ORIENTED REQUIREMENTS

[1]A. SOUJANYA, [2]SIDDHARTHA GHOSH

[1]M.Tech Student, Department of CSE, Keshav Memorial Institute of Technology(KMIT), Narayanaguda, Himayathnagar, Telangana, India.
Jansow506@gmail.com

[2]Professor, Department of CSE, Keshav Memorial Institute of Technology (KMIT), Narayanaguda, Himayathnagar, Telangana, India.

Siddhartha@kmit.in

Abstract: Early aspects focus on the problem domain, representing the goals and constraints of users, customers, and other constituencies affected by a software intensive system. However, analysis of early aspects is hard because stakeholders are often vague about the concepts involved, and may use different vocabularies to express their concerns. In that the Goal modeling fits model-driven engineering (MDE) in that it captures stakeholder concerns and the interdependencies using concepts that are much less bound to the underlying implementation technology and are much closer to the problem languages. Aspect-oriented Programming (AOP) provides language constructs to facilitate the representation of multiple perceptions. Synthesis of AOP and MDE not only manages software complexity but also improves productivity, as well as model quality and longevity. In this paper, we propose a model-driven framework for tracing aspects from requirements. These aspects can be discovered during goal-oriented requirements analysis. This proposal includes a systematic process for discovering aspects from relationships between functional and nonfunctional goals.

Keywords: Discovering Early aspects, Goal-oriented Requirements, Aspect-oriented Programming (AOP),Model-drivenEngineering

I. INTRODUCTION

The marriage of the aspect-oriented (AO) paradigm and business process modeling brings benefits in terms of modularity to the process modeling. According to conventional process modeling notations and languages have some functional decomposition principles and separation of concepts, but they are not enough to represent concepts repeated in the same process or in several different activities and in other processes. Even if a process model is conventionally modularized, several concerns, the so-called crosscutting concerns, remain scattered throughout the process, generating models with reduced understandability and reusability and increased maintenance effort. This lack of crosscutting concerns modularization causes some drawbacks such as a little modification in part of a process will require changes in several parts of the process. Using AOBPM techniques, crosscutting concerns are represented as aspects. This strategy divides the process into core process and aspect process, improving modularity, understandability and maintainability of the process model The transformation of a process model into an as pectized process model involves three steps: (i) identifying crosscutting concerns in the source process model, (ii) transforming these concerns into aspects and iii) modularizing the process with an AO-BPM language. For the first step some heuristics were proposed by Cappelli et al. [2]: "(i) if the concept is repeated several times in different places, (ii) if it is used by different other concepts, (iii) if it reflects an integration of semantically distinct situations, (iv) if it represents a decision situation from which different options may be taken, (v) if its absence does not interfere with the global goals of the whole, (vi) if it can be reused in other domains and (vii) if it is very much independent of other concepts ". One of the heuristics, (v), is related to the process goal, which is the focus of our proposal.

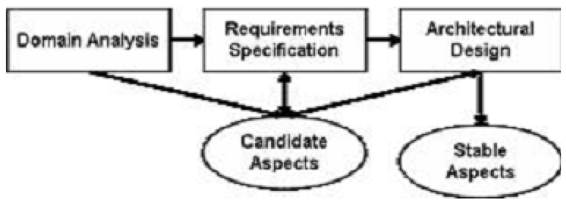

Figure 1 – Early Aspect Scope

We propose a way to use the goal concept in business process modeling to help identify which elements or set of elements are dispensable in order to a process Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee reach its goals. As pointed by the (v) heuristic these elements are considered crosscutting concerns.

II. EARLY ASPECTS

Like other paradigms, such as Procedure Oriented (PO) and OO, and Modular Programming (MP), AO is based upon the separation of concerns (SoC) principle, presented by Dijkstra. Modular programming researchers, as Maynard, have used this principle since the end of 60's to define how many and which are the modules that compose the whole system. The Sock is equivalent to the functional and systems decomposition practices, which comes from the system theory. The aiming of such techniques is to divide the system into smaller parts and treat them in a separated way. Regarding SoC, these concerns after separated, might be implemented as one or more aspects. The EA area lacks a methodology that identify, model and handle aspects on the early phases of a software development process. Building this methodology is the doctoral thesis major contribution of the first author of this paper. Parnas has written one of the most important papers on system decomposition, where criteria based on projects decisions are applied. The EA methodology shall overcome difficulties of applying SoC principle in the requirement specification phase, especially in the specification of Non-Functional Requirements (NFR) which naturally crosscut each other [38] and are scattered and tangled into Functional Requirements (FR). Among concerns involved in software development, crosscutting concerns can be emphasized, which are hard, due to traditional deficiency, to be modeled and implemented by means of scattering and tangling. One of the most important differences of AO deals with that deficiency, aiming to uncouple and encapsulate concerns as security, distribution, synchronization, persistence, mobility, interface, performance, application domain and so on.

III. USING GPM FOR ASPECTS IDENTIFICATION IN PROCESS MODELS

The usual heuristics to identify elements that need to be modularized as aspects were presented at the introduction and most of them are centered on the concept of repetition. However, these heuristics are not enough to identify if an element or a set of elements could become modularized as an aspect. Another important strategy is to identify if the removal of an element does interfere with the global goal. In order to identify elements or set of elements to be modularized as aspects we propose the use of GPM to explicitly represent the initial state and the goals for each process activity, it is related to the sub-domain state. From this explicit representation it is possible to verify if an element can be considered dispensable to the model, in which case it will be considered and modeled as an aspect. To evaluate the importance of a process element with respect to a desired goal, we propose the following procedure:

1. Define the process goals

2. List each activity of the process or set of activities

3. Identify the goals of each item on the list produced in the previous step (they are associated to the output of each activity)

4. For each element in the list and its associated goal, the following analysis must be done:

4.1. Question: if this activity is removed or not executed the goal of the process is still reached? 4.2. Indicate the result of the question (yes or no); 4.3. Add comment related to the answer, this comment allow others to understand the reasons why the activity or set of activities were considered dispensable. 5. Consider all dispensable elements as aspects; 6. Modularize the process using AO-BPM [2]. (*) the process goal may not be specified, but according to Soffer and Wand [6], it is possible to understand the process goal through the analysis of the whole process states and activities. (**) this proposed analysis can be done for single and composed elements, in the case of composed, the goal evaluation must be related to the goal of the entire composition, instead of using each goal of the elements part of the composition. (***) This step is dependent on domain knowledge. The information could be more accurately obtained in the stage of gathering information for further elaboration of the model. It is also possible to get this information from those responsible for this process.

IV. BACKGROUND:

This section aims to situate the existing literature on requirements engineering (RE), MDE, and AOSD. Aspectoriented software development (AOSD) emerged from a rethinking of the relationship between modularization (partitioning software into discrete, non overlapping implementation units) and the time-honored principle of separation of concerns. Any separation-of-concerns criterion leads to a particular partitioning, as though the software were sliced into pieces in a particular direction. Work in aspects has been mostly limited to the implementation phase, dealing with concerns that implementation units have in common, factoring those out as aspects, and employing programming-language-level support to weave the aspects back at loading time, compilation time, or runtime. In fact, AOSD has become nearly synonymous with aspect-oriented programming (AOP) and dominant decomposition usually refers to the system's decomposition into implementation units, such as subsystems, classes, and objects. Most AOSD approaches place the burden for aspect identification and management on the programmer. But crosscutting concerns are often present well before the implementation, such as in domain models, requirements, and the architecture. Dominant decomposition, however, means something different in the early software development activities. A requirements aspect, then, is a concern that cuts across other requirement-level concerns or artifacts of the author's chosen organization. It is broadly scoped in that it's found in and has an (implicit or explicit) impact on more than one requirement artifact. Broadly scoped properties can be quality attributes (nonfunctional requirements) as well as functional concerns that the requirements engineer must describe with relation to other concerns The Requirements has generated a number of notations for modeling stakeholder goals and the relationships between them. Goals express, at various levels of abstraction, stakeholders' many objectives for the system under consideration. Goal-oriented RE uses goal models to elicit, elaborate, structure, specify, analyze, negotiate, document, and modify requirements. Goal modeling shifts the emphasis in requirements analysis to the actors in an organization, their goals, and the interdependencies between those goals, rather than focusing on processes and objects. AOSD applies the principle of separation of concerns to make systems modular so that the intended software is easier to produce, maintain, and evolve. The AOSD community has recognized the importance of considering aspects early on in the software system.

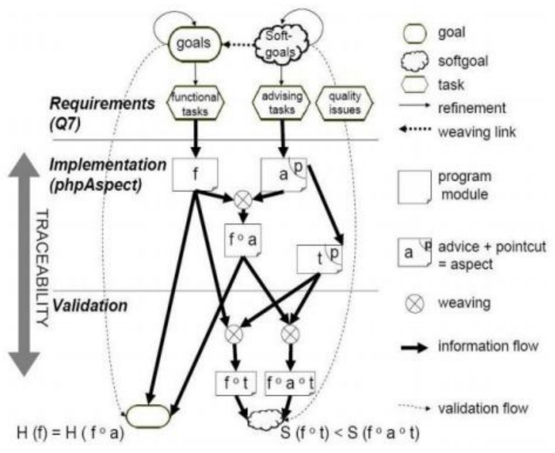

Fig 2:Process overview of the aspect tracing framework

Aspects at the requirements level present stakeholder concerns that crosscut the problem domain. Discovering aspects early can help detect conflicting concerns early, when trade-offs can be resolved more economically Aspect-oriented concepts are modeled explicitly in requirements at the beginning of the development process. Advising tasks, which operationalize soft goals and relate to hard goals, are modularized as aspects and weaved into the goal model to enable aspect-oriented requirements analysis? Goal modeling has become a central activity in RE. It shifts the emphasis in requirements analysis to the actors within an organization, their goals, and the interdependencies between those goals, rather than focusing on processes and objects

V. CONCLUSION

The results of the workshop show that the early aspects topic is still in its infancy but progressing. The goal of the workshop was primarily not to find solutions but first to identify the right questions to shape the research of the early aspects topics. During the workshop a number of key research areas have been identified to scope and consolidate the area of early aspects. Currently we can state that the scope of the early aspects domain is defined to a large extent. This workshop showed that several ideas are recurring with respect to the previous early aspects workshops. In particular there seems now to be an agreement that early aspects refers to the aspects that can be identified during the requirements analysis, domain analysis and architecture design phases. This means that aspects during the detailed design are not counted as early aspects. In this report we have termed the aspects at the detailed design as intermediate-aspects. Aspects which refer to the crosscutting concerns at the implementation phase, testing and maintenance phases are termed as late aspects. Since early aspects refers to the crosscutting concerns during the requirements analysis, domain analysis and architecture design phases, the research is also focused on these three phases. So far, in general, the research on early aspects appeared to proceed separately and independently in each of these phases. It appears, however, that the early aspects in the three phases are not independent and directly impact each other. A concern such as synchronization that is identified during the requirements analysis phases requires the modeling of it during the architecture design. During the domain analysis some aspects might be identified which were overlooked during the requirements analysis phases. There is certainly a relation among the concerns but so far the parity and the semantics of the relations among the concerns in the early phases are not completely clear yet and more research is required to crystallize the concepts. This workshop and the previous workshop have shown that early aspects exist and that they need to be handled with care to provide better maintainable software. In the future, we expect that the questions addressed in this workshop will be solved gradually.

REFERENCES:

[1] Ghattas, J., Soffer, P., Peleg, M. Learning Business Process Models: A Case Study. BPM 2007 Workshops (LNCS 4928), p. 383-394. 2008.

[2] Scheer, A-W., Nüttgens, M. ARIS Architecture and Reference Models for Business Process Management. In W. van der Aalst et al. (Eds.): Business Process Management, LNCS 1806, pp 376-389, Springer-Verlag Berlin Heidelberg. 2000

[3] A. Finkelstein. The viewpoints faq. BCS/IEE Software Engineering Journal, 11(1), 1996.

[4] J. Grundy. Aspect-oriented requirements engineering for component based software systems. In 4th IEEE International Symposium on Requirements Engineering, pages 84– 91.

[5] M. Katera and S. Katz. Architectural views of aspects. In Proceedings of the International Conference on Aspectoriented Software Development, pages 1–10, 2003.

[6] A. Rashid, A. Moreira, and J. Araujo.Modularisation and composition of aspectual requirements. In Proceedings of the International Conference on Aspect-oriented Software Development, pages 11–20, 2003.

[7] L. Liu, E. Yu, and J. Mylopoulos.Security and privacy requirements analysis within a social setting. In RE 2003, pages 151–161, 2003.

[8] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. IEEE Transactions on Software Engineering, 18(6):483–497, June 1992.

[9] J. Mylopoulos, L. Chung, and E. Yu.From object-oriented to goal-oriented requirements analysis. Communications of the ACM, 42(1):31–37, Jan. 1999.

[10] pair Networks. oscommerce: Open Source E-Commerce Solutions, http://www.oscommerce.com.

[11] A. Rashid, P. Sawyer, A. M. D. Moreira, and J. Arajo. Early aspects: A model for aspect-oriented requirements engineering. In RE 2002, pages 199–202, 2002