Tracing and Validating Goal Aspects

Yijun Yu
The Open Univ., UK
y.yu@open.ac.uk

Nan Niu
Univ. of Toronto, Canada
nn@cs.toronto.edu

Bruno González-Baixauli
Univ. of Valladolid, Spain
bbaixauli@infor.uva.es

William Candillon
Univ. of Lille, France
wcandillon@elv.enic.fr

John Mylopoulos
Univ. of Toronto, Canada
jm@cs.toronto.edu

Steve Easterbrook
Univ. of Toronto, Canada
sme@cs.toronto.edu

Julio Cesar Sampaio do Prado Leite
PUC-Rio, Brazil
julio@inf.puc-rio.br

Gilles Vanwormhoudt
Univ. of Lille, France
vanwormhoudt@enic.fr

Abstract

Aspects promote a clear separation of concerns so that tangled and scattered concerns are modularized throughout software development. We propose a framework to trace aspects identified during goal-oriented requirements analysis to code and testing. Two types of checks are performed to validate the resulting system in light of stakeholders' crosscutting concerns. One ensures that systems with and without aspects have the same functionality defined by the hard goals. The other checks whether the weaved system with aspects indeed improves system qualities in terms of the degree of softgoal satisfaction. We demonstrate the approach using an open-source e-commerce platform.

1 Introduction

Aspect-oriented programming (AOP) modularizes crosscutting concerns so that programs' evolution is less problematic [11]. Uncovering aspects from system implementation can be supported by using program analysis techniques such as aspect mining and refactoring [17]. A common limitation to these techniques is that it is hard to validate the modularized code aspects against their very purposes of existence: Are they required or designed to be there?

Thus, early aspects – the concerns that crosscut an artifact's dominant separation-of-concern criterion in the early stages of the software life cycle [3] – are studied to help the stakeholders and designers consider aspects early on, before these crosscutting concerns start to clutter the code artifacts [15, 16, 20].

Although various approaches have been proposed to discover and modularize aspects at the requirements level [1], few address the validation and traceability issues of the identified early aspects. Specifically, it is of great interest to trace how early aspects manifest themselves to code and testing, and to validate the resulting system in light of the stakeholder concerns that crosscut the problem domain.

In our previous work [20], we developed a framework for identifying and weaving candidate aspects during goal-oriented requirements analysis [18]. Goal aspects are captured as the operationalizations of softgoals and the relations to functional goals. They are suited to be implemented as code aspects, but developers may choose other means to address these crosscutting concerns. Even in the latter case, it is desirable to keep early aspects modularized so that one does not have to recover them from the code at a later date.

In this paper, we propose a framework to trace and validate the identified goal aspects [20]. Early aspects are either naturally mapped to code aspects, or recorded as issues to directly advise testing. Aspect testing, therefore, is guided by stakeholder goals. In our approach, aspects are intended to enhance system qualities by interacting with multiple system units, while preserving the functionalities defined by hard goals.

The goals of our work are twofold. By separating crosscutting concerns throughout requirements, implementation, and testing phases, we achieve a high degree of modularity and traceability in software development. By validating implementation against stakeholder concerns, we achieve a high level of software quality and user satisfaction. To those ends, this paper provides several contributions:

1. We propose an approach to tracing goal aspects to implementation and testing;
2. We define criteria and means of measuring aspects as separate metrics to be weaved into base modules; and
3. We present the preliminary results of applying the approach to an open-source platform written in PHP.

2 Aspects in goal models

A requirements aspect is a stakeholder concern that cuts across other requirement-level concerns or artifacts of the

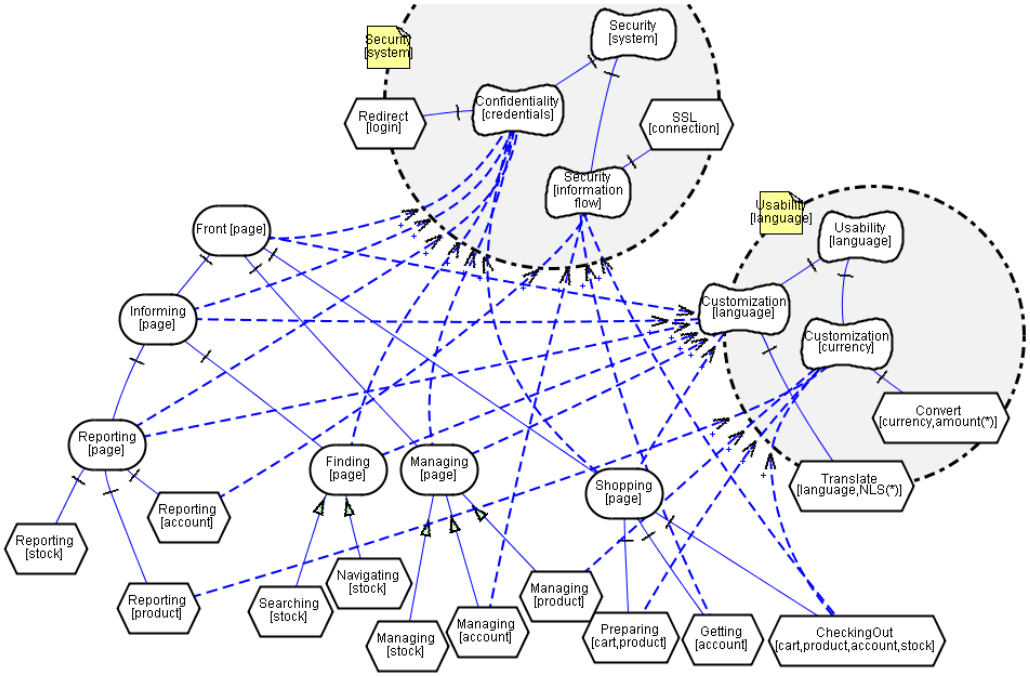

Figure 1. Illustration of goal aspects in media shop using i^* notations [19]

author's chosen organization [3]. To interpret the notions involved in aspect orientation from a requirements perspective, we follow the metaphor that every requirements aspect acts as a service provider to some base modules [13].

- **Advice** defines the content of the service that an aspect provides. It describes *what* the service is about.
- **Join points** are points in the base with which an aspect interacts. They describe *where* the service is provided.
- **Pointcut** represents a set of join points. It describes the *situational patterns* of an aspect's service.
- **Weaving** is the process of coordinating service providers (aspects) and consumers (base requirements). It describes *when* and *how* the service occurs.

In addition, the purpose of a requirements aspect describes *why* the service is needed in the first place. Since identifying aspects too early is counterproductive [14], some approaches deal with structured models, such as goal graphs [20] or problem frames [9].

Organizational goals lead to requirements. Goals justify and explain the presence of requirements, while providing the baseline for validating stakeholder concerns [18]. Goal modeling frameworks distinguish between *hard (functional) goals* – states that actors can attain – and *softgoals*, which can be satisfied only to certain degrees. Aspects in goal models can be discovered using the correlations from hard goals to softgoals along with a goal eliciting and refinement process of a V-shape goal graph [20].

As a running example, Figure 1 shows two goal aspects – security and usability – in the media shop study [5, 20]. The top level softgoals are captured as goal aspects. The aspect weaving is achieved by composing the advising tasks with the functional tasks of effected hard goals. Such a weaving helps remove the dotted arrows in Figure 1 so that scalability of the goal model is improved. As an example, the aspect "Customization [language]" is operationalized into an advising task "Translate [language, NLS]", meaning that the media shop is advised to translate occurrences of natural language strings (NLS) into the desired language. This advice crosscuts all hard goals that display Web pages. Basic functionalities (e.g., "Informing" and "Reporting") defined by hard goals via functional tasks shall not be changed, though, by weaving such a usability aspect.

3 An aspect tracing and validating framework

Figure 2 shows the process of aspect tracing and validating. Advising tasks, shown in the upper part of the figure, are modularized and weaved into goal models.

Key concepts of AOP implementation are depicted in the middle of Figure 2. Functional modules (f) and code aspects (advice + pointcut) are derived from functional and advising tasks respectively. The weaved system ($f \circ a$) is

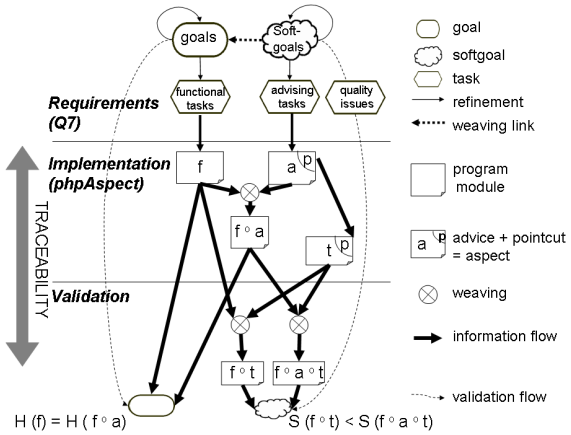

Figure 2. Process overview

Table 1. Tracing the security (𝕊) and usability (𝕌) aspects in an osCommerce media shop

Concept	Q7	phpAspect	Softgoal Validation
aspect (𝕊)	`<aspect>::Security [system]`	`aspect Security`	Use PHPUnit to generate unit testing aspects to verify http authentication and page redirection. Validation result: security ensured.
pointcut (𝕊)	`<= + * [page]` `<= + * [cart]`	`call(Page->goTo($arg2))` `exec(Cart->*(*))`	
advice (𝕊)	`{ & Redirect [login] }` `{ & SSL [connection] }`	`checkCredentials{...}` `checkSSL{...}`	
aspect (𝕌)	`<aspect>::Usability [language]`	`aspect Usability_Language`	Use both the pspell testing harness and a Spanish tester to check the correctness of language translation, date display format, and currency conversion. Validation result: usability enhanced.

pointcut (𝕌)	`<= + * [page]`
	`<= + * [date]`
	`<= + * [amount]`
		`call(Page->*printf(*))`
		`call(Data->strftime($arg2))`
		`exec(Amount->display($arg2))`	

advice (𝕌)	`{ & Translate [language, NLS] }`
	`{ & Display [format, date] }`
	`{ & Convert [currency, amount] }`
		`translatePage{...}`
		`dateTimeFormat{...}`
		`convertCurrency{...}`	

obtained by composing advice (a) with bases according to the pointcut description (p). Some early aspects may not be mapped to code at all. For example, a typical performance requirement might state that the system shall complete a task within 2 seconds. These aspects play a key role in monitoring the system's behavior, and shall not be lost in software development. We record them as quality issues to enable traceability, but the discussion of handling these quality issues is beyond the scope of this paper. The success criteria for aspects are specified in t, which gathers quality metrics and shares the same pointcut with a. It is important to incorporate the metrics t so that one can measure system qualities with ($f \circ a \circ t$) and without ($f \circ t$) aspects.

System validation is shown in the lower part of Figure 2. The weaved system ($f \circ a$) is subject to two tests. The first test ensures that systems with and without aspects have the same functionality defined by hard goals: $H(f) = H(f \circ a)$. Existing testing mechanisms, such as unit testing, can be reused to validate whether the weaved system satisfies the functional requirements. The second test checks whether the weaved system indeed improves system qualities in terms of the degree of softgoal satisfaction: $S(f \circ t) < S(f \circ a \circ t)$. Our approach enables both forward and backward tracing of crosscutting concerns throughout the software life cycle.

Table 1 presents the preliminary results of applying the framework to osCommerce [2], an open-source platform on which media shop development can be based. Goal aspects are represented in Q7 [12], a pseudo programming language that captures the structure of requirements goal graphs, while incorporating aspect orientation. In Table 1, the security (𝕊) goal aspect redirects unauthorized users to the login page, and requires all shopping cart operations to be performed in an SSL mode. Usability (𝕌) is AND-decomposed into 3 parts: natural language strings (NLS) translation, date formatting, and currency conversion.

We aim to equip developers with a competent framework to implement goal aspects as code aspects. We worked out phpAspect [4], a solution for AOP in PHP, where both the source program and the aspect modules are XMLized into abstract syntax trees, which are weaved through customized XSLT stylesheets. The woven XML syntax trees are transformed into the target PHP program via unparsing XSLT stylesheets [21]. In Table 1, the traces between goal aspects and code aspects can be readily spotted. Specifically, we map goal's topics into parameterized pointcuts, and map softgoal's operationalizations into advices. Although a one-to-one correspondence exists between the name of a goal aspect and that of a code aspect in Table 1, many-to-many mappings are expected and more advanced traceability mechanisms need to be sought in more general cases.

Validation in our approach ensures that system functionalities are preserved and system qualities are enhanced by weaving aspects into base modules. The unit test cases of the functional tasks can be reused without any change for checking the functionalities of the weaved system. For example, the shopping cart sum computation must be the same regardless of which natural language being used by the media shop customer. On the other hand, certain qualities in a system with weaved aspects must outperform the one without aspects, so that the effort of managing aspects can be justified. Measuring quality attributes typically presents an obstacle to traditional testing mechanisms, since softgoals are not always easy to metricize. Our effort of modeling aspects early in the requirements pays off here. The results from goal-oriented analysis, including the quality metrics, the advising task and pointcut of goal aspects, can be reused and extended to check the degree of softgoal satisfaction. Key steps and initial results of validating goal aspects from the running example are highlighted in Table 1. It is our ongoing work to carry out automated tests more thoroughly.

4 Discussion and conclusions

Grundy [8] and Rashid *et al* [16] are among the earliest who recognize the advantage of aspects for reasoning about component-based software at the requirements level.

In comparing the related work listed in a literature survey of early aspects [13], our key contribution brings early aspects' traceability links and validation together. In [6], traces between early aspects and designs are set up by comparing naming conventions, term co-occurrences, etc. It is not clear how traceability through such queries can be verified, as the leap from design to implementation may distort the precision. Our work ameliorate the problem in that we explicitly rely on the decomposition and contribution links in goal models to build traceability among goals and aspects.

The work of Cleland-Huang *et al.* [7] extracts non-functional requirements (NFRs) based on information retrieval techniques. The strength of traceability is quantified as precision and recall of the keyword-based search. When naming conventions mismatches the functionality specified in the program, our approach may improve the precision through the semantics of executed test cases.

In [10], proof obligations were introduced to formalize the validation of aspectual requirements in programs with well-defined axiomatic semantics. For the quality attributes that do not have a clear-cut answer to satisfaction, it is necessary to validate whether and how much the system can be improved after weaving the aspects. For example, instead of proving a word is Spanish, we showed how well a Spanish user understood it. We believe our approach complements proof obligations in validating requirements aspects.

In this paper, we have proposed a framework to investigate how the aspects discovered from requirements goal models can be validated and traced throughout the software life cycle. We assumed only NFRs are traced into aspects, and aspects' weaving does not change base module's functionality defined by the hard goal. We presented the initial investigation of the approach to reengineering a public domain e-commerce platform. Our work also verified the initial AOP claim: It is natural to implement the globally concerned NFRs as aspects that cut across the subsystems [11].

Our future work includes thoroughly checking the goal aspects in osCommerce so that the scalability issue can be addressed. Also of interest would be extending the approach to incorporate with code aspects that are derived from functional requirements, and goal aspects that manifest themselves in other forms than code aspects, such as quality issues, functions, design decisions, and the like.

Acknowledgments. *We would like to thank invaluable discussions and feedback from our colleagues: Robin Laney, Bashar Nuseibeh at the Open University, and Eric Yu, Rick Salay at the University of Toronto.*

References

[1] Early aspects portal. *Available at http://www.early-aspects.net*, Last accessed on May 30, 2007.

[2] osCommerce. *Available at http://www.oscommerce.org*, Last accessed on May 30, 2007.

[3] E. Baniassad, P. C. Clements, J. Araújo, A. Moreira, A. Rashid, and B. Tekinedoğan. Discovering early aspects. *IEEE Software*, 23(1):61–70, Jan/Feb 2006.

[4] W. Candillon. Goodle summer of code. *Available at http://code.google.com/soc/php/about.html*, Last accessed on May 30, 2007.

[5] J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: The Tropos project. *Information Systems*, 27(6):365–389, 2002.

[6] S. Clarke and E. Baniassad. *Aspect-Oriented Analysis and Design: The Theme Approach*. Addison-Wesley, 2005.

[7] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *Intl. Requirements Engineering Conf.*, pages 39–48, 2006.

[8] J. C. Grundy. Aspect-oriented requirements engineering for component-based software systems. In *Intl. Symp. on Requirements Engineering*, pages 84–91, 1999.

[9] C. B. Haley, R. C. Laney, and B. Nuseibeh. Deriving security requirements from crosscutting threat descriptions. In *Intl. Conf. on Aspect-Oriented Software Development*, pages 112–121, 2004.

[10] S. Katz and A. Rashid. From aspectual requirements to proof obligations for aspect-oriented systems. In *Intl. Requirements Engineering Conf.*, pages 48–57, 2004.

[11] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *European Conf. on Object-Oriented Programming*, pages 220–242, 1997.

[12] J. Leite, Y. Yu, L. Liu, E. Yu, and J. Mylopoulos. Quality-based software reuse. In *Intl. Conf. on Advanced Information Systems Engineering*, pages 535–550, 2005.

[13] N. Niu, S. Easterbrook, and Y. Yu. A taxonomy of requirements aspects. In *Early Aspects Wkshp at AOSD*, 2007.

[14] B. Nuseibeh. Crosscutting requirements. In *Intl. Conf. on Aspect-Oriented Software Development*, pages 3–4, 2004.

[15] A. Rashid, A. Moreira, and B. Tekinerdogan. Early aspects: aspect-oriented requirements engineering and architecture design (guest editorial). *IEE Software*, 151(4):153–155, 2004.

[16] A. Rashid, P. Sawyer, A. Moreira, and J. Araújo. Early aspects: A model for aspect-oriented requirements engineering. In *Intl. Requirements Engineering Conf.*, pages 199–202, 2002.

[17] A. van Deursen, M. Marin, and L. Moonen. Aspect mining and refactoring. In *Wkshp on Refactoring: Achievements, Challenges, Effects at WCRE*, 2003.

[18] A. van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Intl. Symp. on Requirements Engineering*, pages 249–262, 2001.

[19] E. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Intl. Symp. on Requirements Engineering*, pages 226–235, 1997.

[20] Y. Yu, J. S. P. Leite, and J. Mylopoulos. From goals to aspects: discovering aspects from requirements goal models. In *Intl. Requirements Engineering Conf.*, pages 38–47, 2004.

[21] Y. Yu et al. osCommerce's phpAspect report. *Available at http://www.cs.toronto.edu/~yijun/aspectPHP*, Last accessed on May 30, 2007.