A Probabilistic Approach to Evaluating Early Aspects

Chuan Duan, Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou
DePaul University
School of Computer Science, Telecommunications and Information Systems
{duanchuan, jhuang, rsettimi, xzou}@cs.depaul.edu

Abstract

The paper proposes a semi-automated approach for supporting decision making during early aspect identification. A probabilistic trace retrieval tool is used to dynamically generate links between potential aspects initially identified through subgoals and operationalizations in a Softgoal Interdependency Graph, and artifacts representing early functional descriptions of the system. The user evaluates the links to determine if they impact the system as cross-cutting concerns that could be modeled as aspects. An example is used to illustrate the proposed approach.

1. Introductory

Aspects are described as "crosscutting concerns" representing functionality that is interwoven throughout the system, and that ultimately can lead to the proliferation of scattered and tangled code. At the requirements and architecture level these cross-cutting properties are commonly referred to as early aspects. Examples include confidentiality, fault tolerance, security, and real time constraints. By identifying aspects at an early stage it is possible to design a well-modularized system that incorporates aspects as part of the initial design, rather than through later identification and refactoring. This approach can lead to better design decisions and can reduce the cost and effort involved in building the system.

Several researchers have investigated the problem of early aspect identification. Rashid et al [1, 2] described the Aspect-Oriented Requirements Engineering (AORE) model for defining component and requirements-level aspects. Based on domain knowledge of the developer, concerns are identified and described at an early stage. In AORE, an aspect is identified by determining the extent to which it cuts across several viewpoints (See [3] for details of viewpoints).

Baniassad and Clarke [4] proposed the use of a themed approach in which functionality is categorized either as a base theme or a cross-cutting theme. They use tool support to evaluate a set of requirements in order to identify related themes and to differentiate between erroneous, coincidental, hierarchical, or cross-cutting relations. Their tool provides extensive support for aspect identification but requires a significant level of user input.

This paper also addresses the issue of early aspect identification by proposing the use of a probabilistic trace retrieval tool to provide semi-automated support for identifying early aspects. The tool, known as Poirot:Tracemaker [8] has the ability to dynamically generate traceability links and is used to evaluate the cross-cutting impact of potential aspects that are initially identified through examining the subgoals of a Softgoal Interdependency Graph (SIG). Poirot returns information to the user in the form of a list of impacted artifacts that

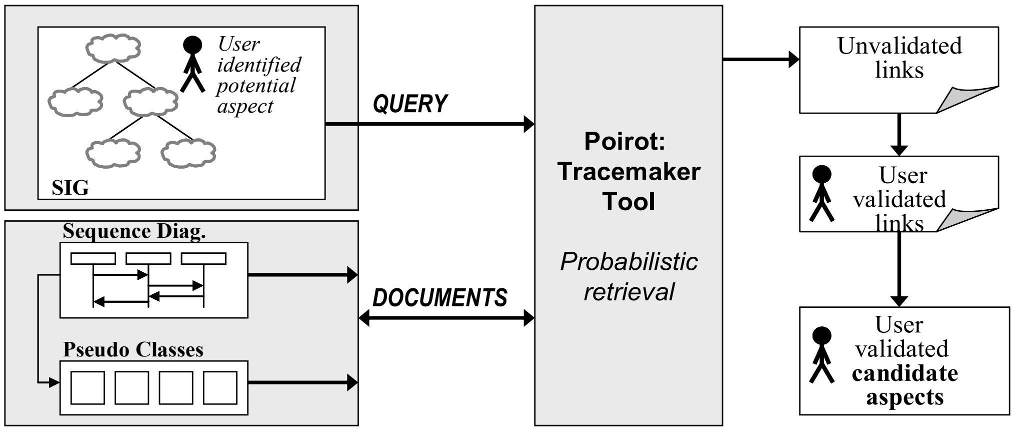

Figure 1. Poirot traceability support for early aspect identification.

the user can evaluate to determine the cross-cutting impact of a potential aspect. The approach is summarized in Figure 1.

The following section provides a brief survey of our approach discussing two alternate options for identifying cross-cutting concerns. We discuss the approach using an example drawn from the requirements and sequence diagrams of an Ice Breaker System (IBS) described in [8, 9]. The paper concludes by discussing key issues and future work.

2. Mining potential aspects

The term "candidate aspect" has been used to represent an early aspect that may or may not be modeled as an aspect in the later design of the system. In this paper we use the term "potential aspect" as shorthand for a "potential candidate aspect." A potential aspect represents a potential cross-cutting concern that triggers a trace query from which the user either confirms or rejects its status as a candidate aspect.

Non-functional requirements and their interdependencies are first modeled as softgoals in a SIG [5]. A SIG contains three primary types of nodes. These are goals, subgoals, and operationalizations (Ops). Goals are decomposed into subgoals describing qualities desired in the system. Each subgoal can be further refined into lower-level subgoals. At the lowest level, subgoals are decomposed into Ops which describe potential design level solutions of the goal. In Figure 2, the leaf nodes representing the Ops are shown in bold.

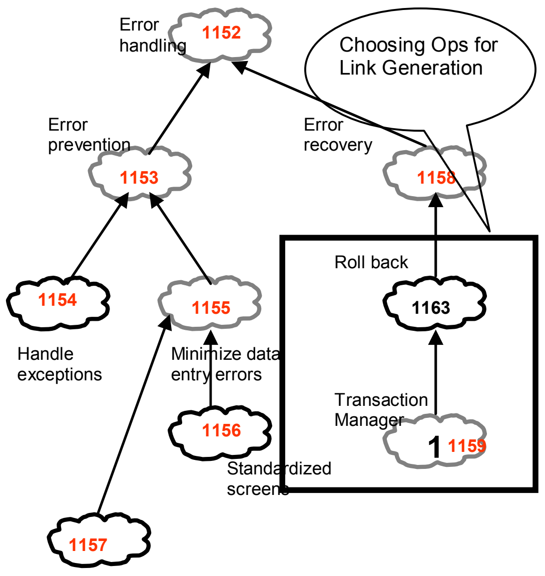

Figure 2. A section of the Ice Breaker SIG showing the transaction manager subgoal.

To identify potential aspects, the user inspects subgoals in the SIG in order to identify ones that intuitively have potential cross-cutting impact across the design of the system. The user then issues a dynamic trace query to determine the set of impacted artifacts.

2.1. Trace generation

In our approach, traces are generated using a probabilistic Information Retrieval model, whose goal is to identify relationships between queries and a collection of documents. The IR tool computes the probability of a link between a query and a document in the searchable document space. Such probability values can be regarded as similarity scores between the query and the document. The model will identify potential traces between a query and a document whenever the probability value of a link is above a certain threshold value.

Almost all types of system elements, design diagrams, code, and test cases, may play the role of either a query or a document. With aspect identification, a SIG subgoal and its descendents act as the query and use cases, sequence diagrams, requirements, or other early forms of artifacts serve as the searchable documents.

The formula for calculating the probability between artifact *dj* and query *q* is given as [6, 7]:

$$pr(d_j \mid q) = \left[\sum_i pr(d_j \mid t_i) \mid pr(q \mid t_i) pr(t_i)\right] / pr(q)$$

where t_i denotes an index term It expresses the degree of coverage for document d_j provided by q.

$$pr(t_i) = 1/n_i [\sum_i 1/n_i]^{-1}$$

with n_i is the number of documents that contain the term t_i. The expression of $pr(t_i)$ assumes that the probability of t_i is inversely proportional to the number of documents containing it.

$$pr(d_j \mid t_i) = \frac{freq(d_j, t_i)}{\sum_k freq(d_j, t_k)}$$

This expression can be regarded as the degree of belief that the index term t_i supports the knowledge in d_j. The same explanation applies to

$$pr(q \mid t_i) = \frac{freq(q, t_i)}{\sum_k freq(q, t_k)}.$$

$pr(q) = \sum_i pr(q \mid t_i) p(t_i)$ is the probability of a query q.

An example of utilizing this model to retrieve traces is described in our prior work [8].

2.2. Identifying cross-cutting concerns

The Poirot query returns a set of potentially impacted artifacts which the user has to evaluate to confirm whether the links are genuine. This evaluation is necessary due to the imprecision of any dynamic retrieval approach. Poirot tends to return precision rates of 10-50% at recall levels of 90%. Although manual evaluation is required, dynamic trace methods significantly reduce the effort that would be required by an entirely manual analysis. The user therefore accepts or rejects each returned link, resulting in a list of validated links.

The user then examines the validated lists to determine the extent to which the potential aspect cross-cuts multiple design elements. The general idea is that a true aspect would impact multiple distinct design elements. Although aspects crosscut multiple types of elements, this paper focuses on identifying aspects during the early design stage.

3. Case study

The approach is illustrated through use of SIGs and sequence diagrams taken from the Ice Breaker System. The functional requirements of the system were modeled through the use of twenty sequence diagrams, and the non-functional requirements through nine distinct SIGs representing the goals of accuracy, availability, completeness, cost, extensibility, operational security, performance, safety, and usability. These SIG were composed of 82 goals and subgoals, and 98 operationalizations.

In this section, two different approaches are discussed. Although both provide decision making support they have the ability to identify cross-cutting concerns at different stages of the software development lifecycle.

3.1. Linking between SIG subgoals and sequence diagrams

Traces can be established dynamically between the potential aspect and sequence diagrams or use cases that have been initially developed to represent the functional requirements of the system.

Table 1 depicts the manually validated links which are chosen from nine Poirot generated candidate links for the operationalization of "Transaction Manager" which is a descendant of subgoal "Availability". Since the Op traverses several sequence diagrams, it should be considered as an aspect. The result of the analysis of this case is quite consistent with the manual observation.

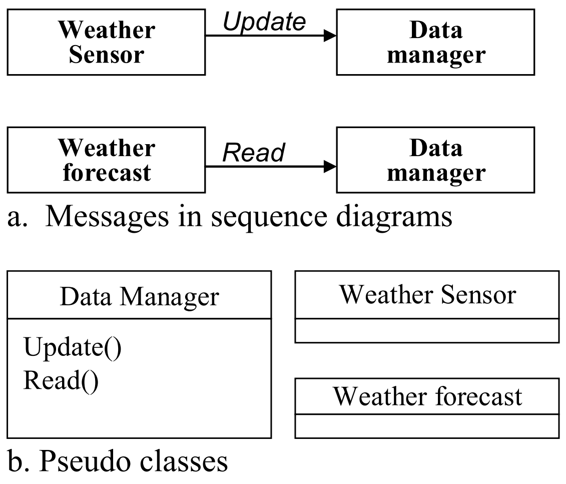

Figure 3. Creation of pseudo classes.

3.2. Linking between SIG and shadow classes

The weakness of the previous approach is that even if a potential aspect crosscuts several sequence diagrams, it may not necessarily crosscut more than one distinct design element. The links to each of the sequence diagrams could have been generated based on the occurrence of similar messages sent between identical classes appearing in the sequence diagrams. If this were the case it would not indicate the need to create an aspect because the functionality could be easily modularized within the two classes.

We therefore propose decomposing the sequence diagram into its composite objects, combining the global knowledge of each object obtained from all the sequence diagrams into a set of early classes that we name pseudoclasses. Links could then be generated against

Table 1. Validated links between Op "Transaction Manager" and sequence diagrams

SIG	Probability	Description	Sequence Diagram Description
1159	0.083333333	Transaction Manager	Record New Road Sensor
	0.025	Transaction Manager	Record Weather Station Reading
	0.023809524	Transaction Manager	Record Treated Road
	0.023809524	Transaction Manager	Add Road Map
	0.022222222	Transaction Manager	Update Weather Forecast

these pseudo classes (also named shadow classes) in order to obtain a more accurate picture of the cross-cutting extent of each potential aspect.

This process is modeled in Figure 3, where the text of message Update and that of message Read is composed into pseudo class Data Manager. This is based on the premise that messages passed to an object represent the functionality of that object.

Table 2 depicts the validated traces generated between the potential aspect and these generated classes. Note that the results more clearly indicate the scope of crosscutting concerns than the previous approach.

Table 2. Validated links between Op "Transaction Manager" and shadow classes

SIG	Probability	Description	Shadow Classes Description
1159	0.1	Transaction Manager	Weather Station Input Proxy
	0.035714286	Transaction Manager	Work Order Manager
	0.025	Transaction Manager	Database Manager

In Table 2, after validation, Op "Transaction Manage" has links with three pseudo classes which in this case is the complete set of links returned by Poirot. We argue that these links provide a more accurate picture of the cross cutting impact of the potential aspect, than the links returned to the sequence diagram themselves. Of course if the design were further advanced and actual class diagrams were available, then the cross-cutting query could have been made directly to the classes within the class diagram, rather than to a set of artificially constructed pseudo classes.

4. Conclusion and future work

This paper describes an initial investigation into the use of semi-automated approach for identifying early aspects. The links returned by the traceability tool support the user in determining whether a potential aspect should in fact be treated as a cross-cutting concern. Such early identification of aspects may lead to better design decisions leading to higher modularization and increased system maintainability. Future work in this area will evaluate the approach in a broader range of examples. More importantly we will investigate whether it is possible to automate the process of identifying potential aspects within the subgoals of the SIG and to produce a cross-cutting report that suggests potential aspects to the user.

Our early results from this particular dataset indicated that SIG subgoals returned a varying number of actual links ranging from a single link to eight links. Further work is needed in order to determine whether metrics could be used to differentiate between cross-cutting and non-crosscutting concerns.

5. References

[1] A. Rashid, A. Moreira, and J. Araujo, "Modularisation and Composition of Aspectual Requirements", 2nd International Conference on Aspect-Oriented Software Development, ACM, 2003, pp. 11-20.
[2] A. Rashid,P. Sawyer, A. Moreira and J. Araujo, "Early Aspects: A Model for Aspect-Oriented Requirements Engineering", IEEE Joint International Conference on Requirements Engineering, IEEE Computer Society Press, 2002, pp. 199-202.
[3] A. Finkelstein and I. Sommerville, "The Viewpoints FAQ", BCS/IEE Software Engineering Journal, 1996, 11(1).
[4] E L. A. Baniassad and S. Clarke, "Finding Aspects in Requirements with Theme/Doc", Position Paper for the Early-Aspects Workshop held as part of AOSD 2004, March 2004.
[5] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic, 2000.
[6] S.K.M Wong and Y.Y.Yao, "On Modeling Information Retrieval with Probabilistic Inference", Information Systems, Vol.16, No.3, 1991, pp. 301-321.
[7] S.K.M. Wong and Y.Y.Yao, "A probabilistic inference model for information retrieval", Information Systems, Vol. 16, No. 3, 1991, pp. 301-321.
[8] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhan and S. Christina et al, "Goal Centric Traceability for Managing Non-Functional Requirements", To appear in International Conference on Software Engineering, St Louis, USA, May 2005.
[9] S. Robertson, and J. Robertson, Mastering the Requirements Process, Addison-Wesley, 1999