PAPER

AspectQuery: A Method for Identification of Crosscutting Concerns in the Requirement Phase

Chengwan HE[†a)], ***Member and*** **Chengmao TU**[†b)], ***Nonmember***

SUMMARY Identification of early aspects is the critical problem in aspect-oriented requirement engineering. But the representation of crosscutting concerns is various, which makes the identification difficult. To address the problem, this paper proposes the *AspectQuery* method based on goal model. We analyze four kinds of goal decomposition models, then summarize the main factors about identification of crosscutting concerns and conclude the identification rules based on a goal model. A goal is crosscutting concern when it satisfies one of the following conditions: i) the goal is contributed to realize one soft-goal; ii) parent goal of the goal is candidate crosscutting concern; iii) the goal has at least two parent goals. *AspectQuery* includes four steps: building the goal model, transforming the goal model, identifying the crosscutting concerns by identification rules, and composing the crosscutting concerns with the goals affected by them. We illustrate the *AspectQuery* method through a case study (a ticket booking management system). The results show the effectiveness of *AspectQuery* in identifying crosscutting concerns in the requirement phase.

key words: *aspect-oriented requirement engineering, crosscutting concern, identification of crosscutting concern, aspect composition*

1. Introduction

Crosscutting concern hinders the modularization of software design and implementation, so AOP (aspect-oriented programming) provides a technique which encapsulates the crosscutting concern as the aspect [1]. However, AOP just solves the problems of code scattering and tangling carried by crosscutting concern in implementation phase, such as refactoring of legacy system [4].

AOSD (aspect-oriented software development) emerges as the development of AOP [2]. It advocates the idea applying the concept of aspect-oriented to the entire life cycle of software development. At early phase of AOSD, the AOP language develops rapidly. Many approaches in implementation phase such as reuse of aspect and definition of join-point are proposed by most researchers. However, where are the aspects in implementation phase from? Can the crosscutting concerns in implementation phase be identified in earlier development phases such as requirement or design phase? How are the crosscutting concerns in the requirement phase converted into the aspects in design and implementation phase? Obviously, aspects need to be managed in higher level. These issues are noticed by some researchers and the concept of early aspects is proposed to represent aspect-oriented requirement engineering and architecture design [3]. Aspect-oriented requirement engineering, one of the phases of AOSD, aims at separation of crosscutting concerns and identification of aspect in requirement phase and provides the support for the later phase of software development, such as design and implementation [5].

The representation of crosscutting concerns in requirement specification is various [6], [7]. So identification of early aspects is not easy. In order to identify crosscutting concerns and then make sure the influence scope of them, obviously, the concerns in requirement specification and relations between them need to be modeled.

Goal-oriented requirement engineering is one of the most important methods for requirement elicitation and organization [18]. A goal model is the set of interrelated goal diagrams that have been put together for tackling a particular problem [19]. Generally, goal model uses the AND/OR tree to represent goal decomposition in the early requirement phase [20].

To avoid the disturbance caused by the various representation of crosscutting concerns to the identification problem, we propose the *AspectQuery* method based on goal model. On the basis of summarizing the main factors affecting the identification, we conclude the identification rules. In *AspectQuery*, first, we should build the goal model according to the requirement document and transform the goal model into XML file. Then we can identify the crosscutting concerns in goal model by the rules. Last, the identification results are composed automatically with the goals affected by the crosscutting concerns by XQuery.

AspectQuery bases mainly on the goal model, so it is the complement of other methods based on the goal model [13], [17]. The main contributions of this paper are as follows.

- We conclude the identification rules of crosscutting concerns based on goal model. The main factors affecting identification include soft-goal, decomposition way of goal, type of parent goal, and the dispersity of goal.
- *AspectQuery* can identify crosscutting concerns from soft-goal, user goal and atom goal level respectively. That is to say, *AspectQuery* can identify not only the crosscutting concerns linked to the soft-goal but also the functional crosscutting concerns.
- *AspectQuery* can support directly the aspect

Manuscript received July 5, 2012.
Manuscript revised December 18, 2012.

[†]The authors are with School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan, Hubei, 430073, China.
a) E-mail: chengwanhewit@126.com
b) E-mail: tcm72913@126.com

DOI: 10.1587/transinf.E96.D.897

composition through XQuery. *AspectQuery* can identify the goals affected by the crosscutting concern and then compose them.

This paper is organized as follows. Section 2 gives an overview of related work. Section 3 establishes the identification rules of crosscutting concerns by analyzing four kinds of goal decomposition models. Section 4 explains the detailed method about *AspectQuery*. Section 5 gives a case study — a ticket booking management system. The last section gives the conclusions.

2. Related Work

Several scholars proposed many approaches to identify crosscutting concerns in the requirement phase and achieved some results. Information retrieval methods [8], [9] identified the common candidate aspect by experience and then searched the whole requirement specification. These methods could identify and model well the non-functional requirement [15].

Chuan Duan et al. described an approach based on hierarchical clustering and an underlying probabilistic algorithm for automating the detection of early aspects [10]. The approach increased the level of automation and minimized the manual effort required by an analyst.

To identify early aspects, E. Baniassad et al. devised the theme approach for modeling the relationships between behaviors in a requirement document, identifying and isolating aspects in the requirements. The theme approach provided supports for aspect-oriented analysis and design [11].

V. Abdelzad et al. proposed a formal method based on Petri Nets for identification of aspects. The proposed method defined requirements and concerns in the formal form by Petri Nets [12]. Concerns nets and dependencies between requirement nets modeled the final system. The execution of the model showed crosscutting concerns which were candidate aspects.

Y. Yu et al. proposed a particular type of a goal model called a V-graph [17]. The model consisted of 3 parts, including goal representing the functional requirements, softgoal representing the non-functional requirements, task contributing to the satisfaction of both goal and soft-goal. When the decomposition of goal model was finished, the goal was crosscutting by task if there was a contribution link between task and soft-goal and more than one chain of contribution links between task and goal.

GPRN framework for requirement modeling was proposed [14]. System requirements consisted of functional goals, non-functional goals and operational goals in the goal meta-model. Refinement of goal ended with a collection of operational goals. Operational goal corresponded directly with one process and was functional goal. So crosscutting concern could be one of the functional goals.

C. Zhang et al. began by matching their past experiences in aspect discovery at the code level with a detailed requirements modeling of the same architecture in KAOS [13]. They observed that satisfying OR-decomposed sub-goals in the KAOS model typically leads to tangled implementations and should be implemented in the aspect-oriented manner. B. Noraz compared several methods about aspect mining of legacy system [4]. Because the transition from requirement analysis to design and implement was natural and continuous, so we could deduce that requirement could be modeled to support the identification of crosscutting concern in requirement phase.

In model-driven engineering, model-to-model transformations define mappings between models [22]. There are a number of model transformation languages such as ATL (Atlas Transformation Language) [23] and AGG (Attributed Graph Grammar) [24]. As a rule-based programming language, ATL uses OCL (Object Constraint Language) to describe the constraint. In contrast to ATL, AGG is a graphical model transformation language.

3. Analysis of Goal Model and Crosscutting Concerns

3.1 Goal Model

The goal model with hierarchy is shown in Fig. 1. The highest goal in the model is business goal which describes why the system should be developed.

User goal is the expectation that the user wants from the system and describes what the system could help user to do. But user goal could not be mapped into system behavior and should be refined further. Only atom goals refined from user goals can be mapped into system behavior.

There are three types of relation between operations in implementation phase, such as sequence, selection and loop. But we introduce the notation of goal set to simplify the modeling process and realize the user goal. Goal set consists of user goal and atom-goal that can not be refined more during requirements analysis. There may be goals that are positive to the soft-goal. The soft-goal proposed by the stakeholders is the expectation about the quality, running environment, resource restriction and external interface. As shown in Fig. 1, business goal, user goal, atom goal and soft-goal are represented with hexagon, rounded rectangle, rectangle and oval respectively.

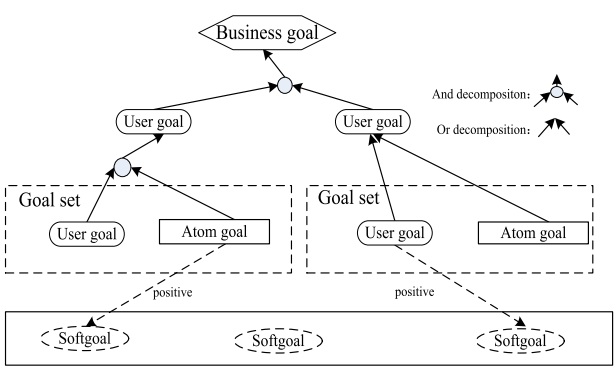

Fig. 1 Goal model.

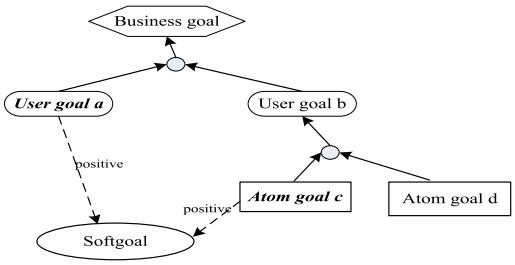

Fig. 2 Phenomenon 1 of goal model.

The definition about the goal model is as follows:

Definition 1: A Goal Model is 2-tuple GM = (G, L), where:

(1) *G is a finite set of goals*, $G \neq \Phi$;
(2) $L \subset (G \times G)$ *is a set of links.*

Definition 2: *g* is one goal of the set *G* with the following properties:

(1) *g.dec* represents the decomposition way of goal *g*. The value set of *g.dec* is {*and*, *or*, *end*}, representing that the *g* is decomposed by *and*, *or* and *end* respectively.
(2) *g.level* represents the level of goal *g*. The value set of *g.level* is {*business*, *user*, *atom*, *soft*}, representing that the *g* is *business goal*, *user goal*, *atom goal* or *soft-goal* respectively.
(3) *g.linkSG* represents whether goal *g* has the link to the soft-goal. The value set of *g.linkSG* is {*yes*, *no*}.
(4) *ParentOf*(*g*) represents the parent goal of goal *g*. The parent goal of *business goal* is empty.
(5) *ChildOf*(*g*) represents the child goal set of goal *g*. The child goal set of *atom goal* is empty.

3.2 Analysis of Goal Model

Because all the goals are refined fully after building the goal model, if the crosscutting concern exists in the goal model, the scatter behavior will be shown in the goal model. Crosscutting concerns can be identified by the dispersity which is the occurrence number of goal as the sub-goal. So the dispersity is the main factor of identification of crosscutting concern. Furthermore, the identification of crosscutting concern has relation with the soft-goal and decomposition way of goal. There exist 4 phenomena of goal model as follows:

(1) A goal in goal model is positive to the soft-goal

As shown in Fig. 2, there exists soft-goal in goal model. From the beginning, the top business goal is affected by the soft-goal. Obviously, the influence would be spread to some user goals or atom goals with the refinement of business goal. As for the goals linked to the soft-goal, *AspectQuery* identifies them as crosscutting concerns. For example, the user goal *a* and atom goal *c* in Fig. 2 are positive to the soft-goal, so both of them are crosscutting concerns.

The formal expression of judgment is as follows:

$$\forall GM,\ G \in GM,\ \ IF\ \exists g \in G \wedge g.linkSG = yes.\ \Rightarrow g \text{ IS } a.$$

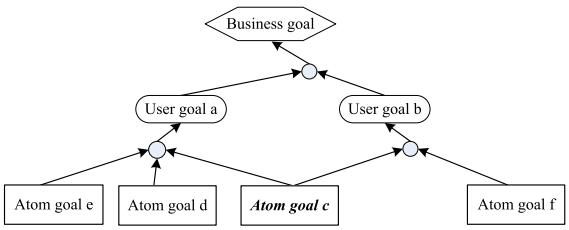

Fig. 3 Phenomenon 2 of goal model.

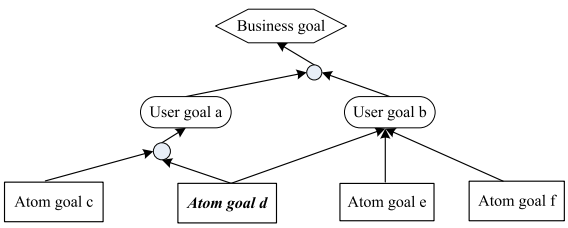

Fig. 4 Phenomenon 3 of goal model.

In the formal expression, *a* denotes the crosscutting concern.

(2) Goal model includes only *and* decomposition

As shown in Fig. 3, goal model is refined by only *and* decomposition. User Goal *a* consists of atom goal *c*, atom goal *d* and atom goal *e* by *and* decompose. User goal *b* consists of atom goal *c* and atom goal *f* by *and* decomposition. Obviously, atom goal *c* influences the implementation of user goal *a*, user goal *b*. So dispersity of atom goal *c* is 2, it is crosscutting concern.

The formal expression of judgment in this phenomenon is as follows:

$$\forall GM,\ G \in GM,\ \exists g_1 \in G \wedge g_1.dec = and,\ \exists g_2 \in G \wedge g_2.dec = and.$$
$$IF\ \exists g \in G \wedge g \in ChildOf(g_1) \wedge g \in ChildOf(g_2).\ \Rightarrow g \text{ IS } a.$$

(3) Goal model includes *or* decomposition

As shown in Fig. 4, goal model includes *or* decomposition. User goal *a* consists of atom goal *c* and atom goal *d* by *and* decomposition. User goal *b* consists of atom goal *d*, atom goal *e* and atom goal *f* by *or* decomposition. Obviously, atom goal *d* influences the implementation of user goal *a*. But identifying atom goal *d* as crosscutting concern depends on the implementation of user goal *b*. If user goal *b* is implemented by atom goal *d*, the dispersity of atom goal *b* is 2, it is crosscutting concern. But if not, it is not. As for the phenomenon of goal like atom goal *d*, if the goal exist the probability of crosscutting concern, *AspectQuery* identifies it as candidate crosscutting concern.

The formal expression of judgment in this phenomenon is as follows:

$$\forall GM,\ G \in GM,\ \exists g_1 \in G \wedge g_1.dec = and,\ \exists g_2 \in G \wedge g_2.dec = or.$$

IF $\exists g \in G \wedge g \in ChildOf(g_1) \wedge g \in ChildOf(g_2). \Rightarrow g$ is *ca*.

In the formal expression, *ca* denotes the candidate crosscutting concern.

(4) Other phenomena

As shown in Fig. 5, user goal *a* consists of atom goal *d* and user goal *b*. user goal *c* consists of atom goal *e* and user goal *b*. Obviously, user goal *b* influences the implementation of user goal *a* and user goal *c*. So dispersity of user goal *b* is 2 and the child goal of user goal *b* is not less than 2. So as for the phenomenon of goal like user goal *b*, *AspectQuery* identifies it and its children goals as crosscutting concerns. The formal expression of judgment in this phenomenon is as follows:

$\forall GM,\ G \in GM,\ \exists g_1 \in G \wedge g_1.level = user,\ \exists g_2 \in G \wedge g_2.level = user$
IF $\exists g \in G \wedge g.level = user \wedge g \in ChildOf(g_1) \wedge g \in ChildOf(g_2). \Rightarrow g$ and $ChildOf(g)$ are *a or ca*.

On the basis of analysis above, the influence factors of identification based on goal model are summarized in Table 1.

- The relation between goal and soft-goal: the goal linked to soft-goal is crosscutting concern.
- Whether the parent goal is crosscutting concern: if parent goal is crosscutting concern, the children goals of it are crosscutting concerns.
- The dispersity of goal: if the dispersity of goal as sub-goal of *and* decomposition is not less than 2, the goal is crosscutting concern; if the dispersity of goal is not less than 2 and the goal as the sub-goal of *and* decomposition is not more than 1, the goal is candidate crosscutting concern.

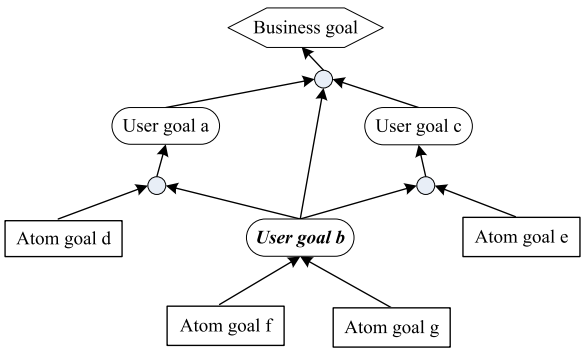

Fig. 5 Phenomenon 4 of goal model.

Table 1 Judgment table for crosscutting concern.

Goal linked to softgoal ?	Parent goal is (candidate) crosscutting concern ?	Dispersity		Judgment result
		Total	And decomposition	
Yes	\	\	\	Crosscutting concern
No	Yes	\	\	(Candidate) crosscutting concern
No	No	>=2	>=2	Crosscutting concern
No	No	>=2	<2	Candidate crosscutting concern

4. AspectQuery

This section introduces detailedly the *AspectQuery* method. An overview of *AspectQuery* is shown in Fig. 6. *AspectQuery* consists of dealing with goal model and dealing with crosscutting concern. Dealing with goal model consists of building and storing goal model. Dealing with crosscutting concern consists of identifying and composing crosscutting concerns.

4.1 Dealing with Goal Model

Dealing with goal model includes building and storing goal model. When requirement specification is got by using existing methods and techniques of requirement elicitation, requirement engineer organizes and decomposes the requirement information by goal model.

In order to identify crosscutting concerns flexibly by XQuery, the goal model should be stored into XML file. The XML schema file should be defined first to reflect the hierarchical structure of goal model. The XML schema file is defined as follows:

```
<? xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<!-- Decompose attribute definition-->
<xs:attribute name="Decompose"
        type="DecomposeType"/>
<xs:simpleType name="DecomposeType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="And"/>
        <xs:enumeration value="Or"/>
        <xs:enumeration value="End"/>
        </xs:restriction>
```

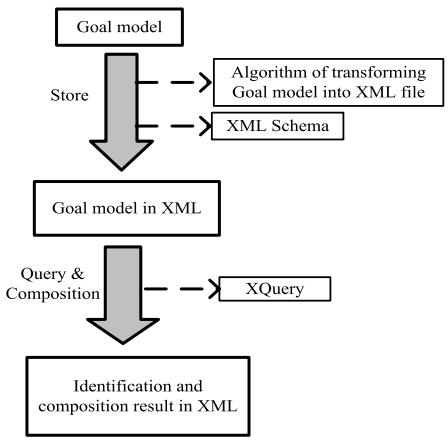

Fig. 6 Overview of *AspectQuery*.

```
</xs:simpleType>
<!--Goal level attribute definition -->
<xs:attribute name="GoalLevel"
        type="GoalLevelType"/>
<xs:simpleType name="GoalLevelType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Business"/>
        <xs:enumeration value="User"/>
        <xs:enumeration value="Atom"/>
        <xs:enumeration value="Soft"/>
    </xs:restriction>
</xs:simpleType>
<!--Softgoal link definition-->
<xs:attribute name="LinkSG"
        type="LinkSGType"/>
<xs:simpleType name="LinkSGType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Yes"/>
        <xs:enumeration value="No"/>
    </xs:restriction>
</xs:simpleType>
<!--Goal element definition-->
<xs:element name="Goal" type="GoalType"/>
<xs:complexType name="GoalType">
        <xs:sequence>
        <xs:element name="GoalName"
                type="xs:string"/>
        <xs:element name="Goal"
            type="GoalType" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="Decompose"
                use="required"/>
        <xs:attribute ref="GoalLevel"
                use="required"/>
        <xs:attribute ref="LinkSG" use="optional"/>
</xs:complexType>
</xs:schema>
```

In XML schema file, the *Goal* tag is defined and has 3 attributes and 2 sub-tags. During storing into the XML file, the *LinkSG* attribute is optional because not all the goals are linked to the soft-goal. Furthermore, the attributes of *Decompose* and *GoalLevel* is required because they have relation with identification level and kind of crosscutting concerns. The *GoalName* sub-tag is used to describe the goal name. The *Goal* sub-tag is used to record the sub-goals of the goal. The *minOccurs* is zero because the Atom goal has no any sub-goals.

The structure of goal model and XML file is the same tree shape, so the goal model could be transformed into XML file according to the depth-first traversal algorithm. The transforming algorithm is as follows:

Procedure: Transform-Goal-Model
Input: rootGoal: rootGoal is the root goal of part or all of a Goal Model
Output: GoalModel.xml: XML file storing the goal model
Precondition: rootGoal $\neq \Phi$ **and** GoalModel.xml $= \Phi$
Postcondition: GoalModel.xml $\neq \Phi$
Begin
    for each g: subGoal **in** rootGoal
        switch(g)
        case *BusinessGoal*:
            Store property and name of g as BusinessGoal into GoalModel.xml;
            Transform-Goal-Model(g);
            break;
        case *UserGoal*:
            Store property and name of g as BusinessGoal into GoalModel.xml;
            Transform-Goal-Model(g);
            break;
        case *AtomGoal*:
            Store property and name of g as AtomGoal into GoalModel.xml;
            break;
        End switch
    End
End

4.2 Dealing with the Crosscutting Concerns

Dealing with the crosscutting concerns includes identifying and composing crosscutting concerns. After identifying crosscutting concerns, the goal influenced by the crosscutting concerns could also be identified and is called Join goal in this paper. Then crosscutting concern and join goal could be composed. The identification and composition should be discussed from the soft-goal and functional goal level respectively.

(1) Identification and composition of crosscutting concern linked to the soft-goals

Because the goal linked to the soft-goal in the goal model has the *LinkSG* attribute. The identification could be realized by checking whether the goal has *LinkSG* attribute. The identification process for goal linked to soft-goal uses the programming language-like notation.

Procedure: Identify-SoftGoal-Aspect
Input: GoalModel goalModel
Output: List aspectList
Precondition: goalModel $\neq \Phi$ **and** aspectList $= \Phi$
Postcondition: aspectList $\neq \Phi$
Begin
    for each g: goal **in** G of goalModel
        if(g.LinkSG == yes)
            then Store g **and** g.dec **into** aspectList;
    End
    return aspectList;
End

The identification process uses the *Store* process which is used to store the discovered crosscutting concern and decompose attribute into the container.

The composition process about the goal linked to the soft-goal is defined as follows.

Procedure: Compose-SoftGoal-Aspect
Input: List aspectList
Output: Void
Precondition: aspectList ≠ Φ
Postcondition: display compose result
Begin
  for each a: aspect **in** aspectList
    for each g: goal **in** G of goalModel
      if(a ∈ ChildOf(g))
        then Display a, g **and** g.dec;
    End
  End
End

The composition process uses the *Display* process which is used to show the *softgoal* and *decompose* attribute. The composition result consists of crosscutting concern and join goal. The composition form is as follows:

<Aspect>
  <Advice LinkSG="Yes">Crosscutting concern</Advice >
  <JoinGoal Decompose =" decomposeValue">Join goal</JoinGoal>
</Aspect>

(2) Identification and composition of functional crosscutting concern

Functional crosscutting concern usually has the scatter characteristic in the goal model, so the dispersity can be used to judge the crosscutting concerns. If the dispersity of a goal is not less than 2, the goal could be dealt as the (candidate) crosscutting concern. Obviously, the dispersity of each goal in goal model should be counted by traversing the goal model to identify crosscutting concern. The identification process for functional crosscutting concern is as follows.

Procedure: Identify-Functional-Aspect
Input: GoalModel goalModel
Output: List aspectList
Precondition: goalModel ≠ Φ **and** aspectList = Φ
Postcondition: aspectList ≠ Φ
Begin
  for each g: goal **in** G of goalModel
    if(g.LinkSG==no && CountDispersity(g) >= 2)
      then Store g **and**
        CountDispersity(g) **into** aspectList;
  End
  return aspectList;
End

This identification process uses the *CountDispersity* process which is used to count the occurrences of a goal in goal model.

The composition process for functional crosscutting concern is as follows.

Procedure: Compose-Functional-Aspect
Input: List aspectList
Output: Void
Precondition: aspectList ≠ Φ
Postcondition: display compose result
Begin
  for each a: aspect **in** aspectList
    for each g: goal **in** G of goalModel
      if(a ∈ ChildOf(g))
      then Display a,
      CountDispersity(a), g **and** g.dec;
    End
  End
End

The composition result consists of crosscutting concern, dispersity and join goal. The composition form is as follows:

<Aspect>
  <Advice Dispersity = "dispersity">Crosscutting concerns</Advice>
  <JoinGoal Decompose = "decomposeValue">Join goal</JoinGoal>
</Aspect>

5. Case Study

This section describes the application of *AspectQuery* in ticket booking management system which provides the booking service for users by ticket booking servicer [21]. The requirement of system is as follows. The system can provide ticket query service, booking service and unsubscribe service for users. The system supports the management of customer integral and the report about the ticket sales by day, week and month. The system should settle with the airline company and bank automatically. Furthermore, the soft-goal of system includes the audit, safety, reliability and using easily.

5.1 Building and Storing Goal Model

The goal model of ticket booking management system is shown in Fig. 7.

Table 2 shows the goal name, goal level, decomposition way and information about linking to the soft-goal of each goal in goal model. From the table, the system has 9 user goals which are realized by part of 20 atom goals. Because the complete XML file after storing the goal model is too long to place in this paper. We explain the structure of XML file with the example of booking user goal in Fig. 8.

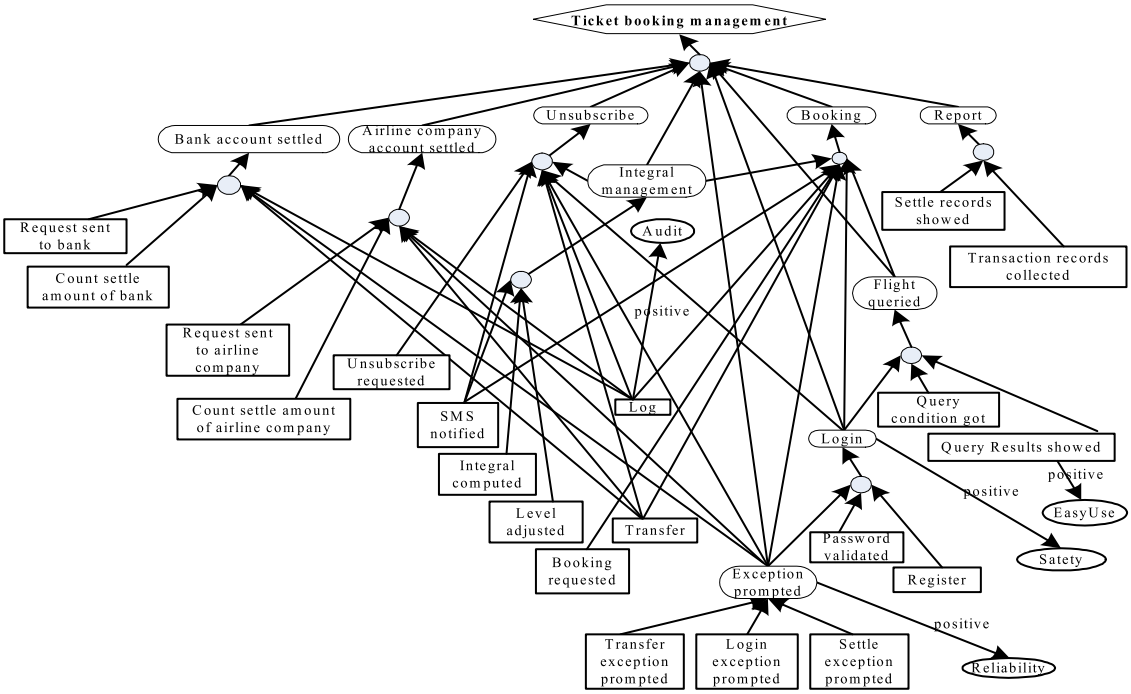

Fig. 7 Goal model of ticket booking management system.

Table 2 Goal information in goal model.

Id	GoalName	Property			
		GoalLevel	Decompose	Child goal Id	NFRLink
1	Ticket booking management	Business	And	2,3,4,5,6,7,8,9,10	N
2	Fight queried	User	And	9,11,12	N
3	Booking	User	And	2,5,9,10,13,14,15,16	N
4	Unsubscribe	User	And	5,9,10,14,15,16,17	N
5	Integral management	User	And	16,18,19	N
6	Bank account settled	User	And	10,14,15,20,21	N
7	Airline company account settled	User	And	10,14,15,22,23	N
8	Report	User	And	24,25	N
9	Login	User	And	10,26,27	Y
10	Exception prompted	User	Or	28,29,30	Y
11	Query condition got	Atom	End	\	N
12	Query results showed	Atom	End	\	Y
13	Booking requested	Atom	End	\	N
14	Transfer	Atom	End	\	N
15	Log	Atom	End	\	Y
16	SMS notified	Atom	End	\	N
17	Unsubscribe requested	Atom	End	\	N
18	Integral computed	Atom	End	\	N
19	Level adjusted	Atom	End	\	N
20	Request sent to bank	Atom	End	\	N
21	Count settle amount of bank	Atom	End	\	N
22	Request sent to airline company	Atom	End	\	N
23	Count settle amount of airline company	Atom	End	\	N
24	Settle records showed	Atom	End	\	N
25	Transaction records collected	Atom	End	\	N
26	Register	Atom	End	\	N
27	Password validated	Atom	End	\	N
28	Transfer exception prompted	Atom	End	\	Y
29	Login exception prompted	Atom	End	\	Y
30	Settle exception prompted	Atom	End	\	Y

```
<Goal Decompose="And" GoalLevel="User">
    <GoalName>Booking</GoalName>
    <Goal Decompose="And" GoalLevel="User" LinkSG="Yes">
        <GoalName>Login</GoalName>
    </Goal>
    <Goal Decompose="And" GoalLevel="User">
        <GoalName>Flight queried</GoalName>
    </Goal>
    <Goal Decompose="End" GoalLevel="Atom">
        <GoalName>Booking requested</GoalName>
    </Goal>
    <Goal Decompose="End" GoalLevel="Atom">
        <GoalName>Transfer</GoalName>
    </Goal>
    <Goal Decompose="End" GoalLevel="Atom">
        <GoalName>SMS notified</GoalName>
    </Goal>
    <Goal Decompose="And" GoalLevel="User">
        <GoalName>Integral management</GoalName>
    </Goal>
    <Goal Decompose="End" GoalLevel="Atom" LinkSG="Yes">
        <GoalName>Log</GoalName>
    </Goal>
    <Goal Decompose="Or" GoalLevel="User" LinkSG="Yes">
        <GoalName>Exception prompted</GoalName>
    </Goal>
</Goal>
```

Fig. 8 Booking user goal in XML file.

5.2 Identifying and Composing Crosscutting Concerns

During dealing with the crosscutting concerns in goal model, the crosscutting concerns of goal linked to soft-goal, goals in user level and goals in atom level can be identified respectively. Because the each identification and composition process is similar, the paper takes the crosscutting concern in user level for example to explain the detailed process of *AspectQuery*. The full code of identifying and composing crosscutting concerns in user level is as follows.

```
declare function local:getUserLevel-Aspect() as element()*
{
    let $aspectThreshold := 2
```

```
<Aspect Dispersity="2">
    <Advice>Flight queried</Advice>
    <JoinGoal Decompose="And">Ticket booking management</JoinGoal>
</Aspect>
<Aspect Dispersity="2">
    <Advice>Flight queried</Advice>
    <JoinGoal Decompose="And">Booking</JoinGoal>
</Aspect>
<Aspect Dispersity="3">
    <Advice>Integral management</Advice>
    <JoinGoal Decompose="And">Ticket booking management</JoinGoal>
</Aspect>
<Aspect Dispersity="3">
    <Advice>Integral management</Advice>
    <JoinGoal Decompose="And">Booking</JoinGoal>
</Aspect>
<Aspect Dispersity="3">
    <Advice>Integral management</Advice>
    <JoinGoal Decompose="And">Unsubscribe</JoinGoal>
</Aspect>
```

Fig. 9 Crosscutting concerns in user-goal level.

```
    let $doc:=doc("GoalModel.xml")
      for $goalName in distinct-values
      ($doc//Goal[@GoalLevel = "User"]//GoalName)
      let $goalNameText:=data($goalName)
      let $dispersity:=count($doc//Goal[@GoalLevel =
        "User"]//GoalName[text()=$goalNameText])
      where $dispersity >= $aspectThreshold
return
<CrosscuttingConcern>
    <GoalName Dispersity=
      "{$dispersity}">{$goalNameText}</GoalName>
</CrosscuttingConcern>
};
<results>
{
let $doc:=doc("GoalModel.xml")
for $aspectConcern in local:getUserLevel-Aspect()
for $goal in ($doc//Goal)
where
some $subGoal in $goal/Goal[@GoalLevel ="User"]
satisfies
data($subGoal/GoalName)=data($aspectConcern)
return
<Aspect Dispersity="{$aspectConcern/GoalName/@Dispersity}">
    <Advice>{data($aspectConcern)}</Advice>
    <JoinGoal Decompose="{$goal/@Decompose}">
            {data($userGoal/GoalName)}</JoinGoal>
</Aspect>
}
</results>
```

Figure 9 shows the results of executing the process above are shown. Table 3 shows the execution results of the process above and the other two processes. Each crosscutting concern record in the table consists of the No., type, dispersity and their join goal Id coming from Table 2. From the Fig. 9 and Table 3, the below phenomena can be found.

The crosscutting concerns linked to the soft-goal includes the exception prompted, log, login and query results showed. The dispersity of exception prompted is 6 and the join goals includes ticket booking management, login, booking, unsubscribe, airline company account settled and bank account settled. The dispersity of log is 4 and the join goals are booking, unsubscribe, airline company account settled and bank account settled. These demonstrate that the dispersity of crosscutting concern linked to the soft-goal is usually not small.

Table 3 Results of three identification process.

No.	Crosscutting concerns	Type	Dispersity	Join goal Id
1	Exception prompted	Usergoal, LinkSG	6	1,3,4,6,7,9
2	Query results showed	Atomgoal, LinkSG	1	2
3	Login	Usergoal, LinkSG	4	1,2,3,4
4	Log	Atomgoal, LinkSG	4	3,4,6,7
5	Integral management	Usergoal	3	1,3,4
6	Flight queried	Usergoal	2	1,3
7	Transfer	Atomgoal	4	3,4,6,7
8	SMS notified	Atomgoal	3	3,4,5

The crosscutting concerns in user level include the flight queried and integral management. The join goal of flight queried is ticket booking management and booking, The join goals of integral management are ticket booking management, booking and unsubscribe.

The crosscutting concerns in atom level include the transfer and SMS notified. The dispersity of transfer is 4 and the join goals are booking, unsubscribe, airline company account settled and bank account settled. The dispersity of SMS notified is 3 and the join goals are booking, unsubscribe and integral management.

6. Conclusions

It is believed that building the requirement model is critical to identify the crosscutting concern in requirement phase. So this paper proposes the *AspectQuery* method based on goal model for identifying and composing the crosscutting concern. While analyzing the existence of crosscutting concern in goal model, we summarize the influence factors of crosscutting concern and conclude the identification rules.

AspectQuery includes mainly 4 steps: building goal model, transforming goal model into XML file, identifying crosscutting concerns and composing crosscutting concern. After storing the goal model into the XML file, *AspectQuery* can implement the identification and composition process automatically by the rules and XQuery. The result of case study demonstrates that the *AspectQuery* can support efficiently for identifying and composing crosscutting concerns in the requirement phase.

However, at present, transforming goal model into XML file is implemented by manually which hinders the full-automation support for identifying crosscutting concern in requirement phase. Hence, we will consider and design the tool for building and transforming the goal model to improve the automation degree of *AspectQuery*. Furthermore, *AspectQuery* can not identify the type of crosscutting concern, that is to say, *AspectQuery* can not point out

whether the crosscutting concern is *before* type or *after* type or even *around* type. Last, some core concerns could have many crosscutting concerns from identification results, such as booking and unsubscribe. In fact, it is necessary to determine the execution order of crosscutting concerns on the same core concern, which is the problem of dealing with conflict [16]. We will research these to improve the proposed method.

Moreover, we will also research about identifying and composing the crosscutting concerns by meta-model of goal model and ATL transformation rules.

Acknowledgements

This research project was supported by the National Natural Science Foundation of China under Grant No. 61272115, 60873024, the Key Science Technology Research Project of Hubei Provincial Department of Education under Grant No.D20121508, and the Excellent Youth Science and Technology Innovation Team Project of the Educational Department of Hubei Province of China (No. T201206).

References

[1] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C.V. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect oriented programming," Proc. ECOOP'97, Number 1241 in Lect. Notes Comput. Sci., pp.220–242, Springer Verlag, 1997.

[2] Aspect-Oriented Software Association. Aspect-oriented software development. AOSD Web page. http://aosd.net/, accessed March 2, 2012.

[3] Early aspects: Aspect-oriented requirements engineering and architecture design. http://www.early-aspects.net/, accessed March 10, 2012

[4] B. Nora, G. Said, and A. Fadia, "A comparative classification of aspect mining approaches," J. Computer Science, vol.2, no.4, pp.322–325, 2006.

[5] J. Grundy., "Aspect-oriented requirements engineering for component-based software systems," 4th IEEE Int'l Symp. on RE, pp.84–91, 1999.

[6] E. Baniassad, P. Clements, J. Araújo, A. Moreira, A. Rashid, and B. Tekinerdoğan, "Discovering early aspects," IEEE Softw., vol.23, no.1, pp.61–70, Jan. 2006.

[7] G. Li, Identification crosscutting concerns in requirement specifications — A case study, Queen's University, Sept. 2009.

[8] L. Rosenhainer, "Identifying crosscutting concerns in requirements specifications," Proc. Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp.24–28, Vancouver, Canada, 2004.

[9] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," CEA'10 Proc. 4th WSEAS international conference on Computer engineering and applications, pp.137–142, 2010.

[10] C. Duan and J. Cleland-Huang, "A clustering technique for early detection of dominant and recessive cross-cutting concerns," presented in Workshops in AORE and Architecture Design, 2007.

[11] E. Baniassad and S. Clarke, "Theme: An approach for aspect-oriented anylysis and design," Int'l Conf. on Software Eng, pp.158–167, 2004.

[12] V. Abdelzad and F.S. Aliee, "A method based on petri nets for identification of aspects," Presented in Information Science and Technologies Bulletin of the ACM Slovakia, vol.2, no.1, pp.43–49, 2010.

[13] C. Zhang, H.A. Jacobsen, and Y. Yu, "Linking goals to aspects," Presented at Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD Conference, 2005.

[14] C. He and C. Tu, "GPRN: A hierarchical framework for aspect-oriented requirement modeling," Int. J. Digital Content Technology and its Applications, vol.5, no.2, pp.165–172, 2011.

[15] D.L.G. Bombonatti and S.S.S. Melnikoff "Survey on early aspects approaches: Non-functional crosscutting concerns integration in software sytems," Proc. CEA'10 of the 4th WSEAS International Conference on Computer Engineering and Applications, pp.137–142, 2010.

[16] L. Guan, X. Li, and H. Hu, "A petri net-based approach for supporting aspect-oriented modeling," 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering, pp.83–90, 2008.

[17] Y. Yu, J.C.S.P. Leite, and J. Mylopoulos, "From goals to aspects: discovering aspects from requirements goal models," Proc. 12th IEEE International Requirements Engineering Conference, pp.38–47, 2004.

[18] A. van Lamsweerde, "Goal-oriented requirements engineering: from system objectives to UML models to precise software specifications," ICSE 2003, pp.744–745, 2003.

[19] Respect-IT sa, Belgium. A KAOS Tutorial. http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf, accessed March 18, 2012.

[20] E.S.K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," RE'97 Proc. 3rd IEEE international Symposium on Requirements Engineering, pp.226–235, 1997.

[21] Q.P. Tan, X.J. Mao, and W. Dong, "Requirement engineering," in Practical tutorial for software engineering, pp.240–251, Higher Education Press, China, 2009.

[22] R. Grønmo, B. Møller-Pedersen, and G.K. Olsen, "Comparison of three model transformation languages," ECMDA-FA 2009, pp.2–17, 2009.

[23] F. Jouault and I. Kurtev, "Transforming models with ATL," Bruel, J.-M. ed. MoDELS 2005. LNCS, vol.3844, pp.128–138, Springer, Heidelberg, 2006.

[24] G. Taentzer, "AGG: A graph transformation environment for modeling and validation of software," Applications of Graph Transformations with Industrial Relevance (AGTIVE), 2003.

Chengwan He received master's degree in Information Engineering of Hokkaido University in Japan. In 2005, he received doctor's degree in Computer Software and Theory of State key laboratory of Software Engineering, Wuhan University, China. His current research interest is theory and application of software engineering.

Chengmao Tu received the B.S. degree in Information and Computing Science from Linyi University in 2009. He now is studying the M.S degree of Technology of Computer Application at Wuhan Institute of Technology. His research interest is software engineer.