Visualizing Aspect-Oriented Goal Models with AoGRL

Gunter Mussbacher, Daniel Amyot
SITE, University of Ottawa
800 King Edward
Ottawa, ON, K1N 6N5, Canada
email: {damyot | gunterm}@site.uottawa.ca

João Araújo, Ana Moreira
CITI/FCT
Universidade Nova de Lisboa
Lisbon, Portugal
email: {ja | amm}@di.fct.unl.pt

Michael Weiss
Department of SCE
Carleton University
1125 Colonel By Drive
Ottawa, ON, K1S 5B6, Canada
email: weiss@scs.carleton.ca

Abstract

As goal models can be large and complex even for small problems, it is often a challenge to aptly visualize them and to efficiently structure them for maintenance and reuse activities. The Goal-oriented Requirement Language (GRL) based on i and the Non-Functional Requirements (NFR) Framework is no exception regarding these challenges. We argue that new ways of visualizing GRL goal models are needed and propose to use Aspect-oriented GRL (AoGRL) to improve the current structure of GRL models and their visualization. The paper presents a case study to evaluate the modularity, understandability, reusability, maintainability, and scalability of AoGRL models compared to GRL models. The evaluation is carried out based on metrics adapted from literature. The evaluation suggests that AoGRL models are more scalable than GRL models and exhibit better modularity, understandability, reusability, and maintainability*

Keywords*: Aspect-oriented Requirements Engineering, Goal-oriented Requirement Language, User Requirements Notation, Aspects, Goal Models*

1. Introduction

Aspects [13] originally provided means for better encapsulation of crosscutting concerns at the implementation-level by addressing the problems of scattering and tangling. Scattering refers to parts of a concern being spread over many classes, while tangling refers to one class containing parts of many different concerns. Recently, more attention has been given to Early Aspects [10] by investigating ways of addressing crosscutting concerns in requirements and design. In particular aspects have been integrated to current requirements engineering approaches such as use cases [11], viewpoints [18] and goals [1]. However work on goals and aspects still needs more investigation.

Goal models are an important part of requirements engineering activities, capturing the rationale, decisions, alternatives, and objectives of the stakeholders of a system. The Goal-oriented Requirements Language (GRL) [4][20][22] is part of the User Requirements Notation (URN) [4][23], a standardization effort of the International Telecommunication Union. It contains two complementary modeling languages for scenarios (Use Case Maps – UCMs) and goals (Goal-oriented Requirement Language – GRL). Like other goal models such as i* [24] and the Non-Functional Requirements (NFR) framework [9], GRL models can be large and complex even for small problems. This paper proposes to use aspect-oriented extensions to GRL, to reduce the complexity of goal models. With *Aspect-oriented GRL* (AoGRL), non-functional requirements, stakeholders, and use cases are visualized as separate concerns in goal models.

Adding support for aspect-oriented modeling to URN presents an excellent opportunity to unify goal-oriented, scenario-based, and aspect-oriented concepts in one framework. Aspects have the potential of improving the modularity, understandability, reusability, scalability, and maintainability of URN models. This paper presents metrics for evaluating such qualities and uses them on a case study to assess the benefits of AoGRL over GRL.

Considering the strong relationship between concerns and non-functional requirements or goals, the Aspect-oriented User Requirements Notation (AoURN) can help bridge the gap between goals and the scenarios which describe how a goal is achieved. Furthermore, concerns in URN models can benefit from a standardized way of modeling with AoURN. Concerns in URN are typically use cases and

goals/non-functional requirements. This paper builds on and complements Aspect-oriented Use Case Maps (AoUCM) [15]. With the help of AoUCM, crosscutting concerns in scenario-based models can already be modeled in an aspect-oriented way. These concerns and others that cannot be expressed with AoUCM can now be linked, justified and analysed from a different angle with AoGRL models.

The remainder of the paper gives an overview of GRL in section 2 and introduces AoGRL in section 3. The case study itself is also briefly discussed in these two sections. Section 4 presents the evaluation of the case study, comparing modularity, understandability, reusability, maintainability, and scalability of AoGRL models with GRL models. Section 5 summarizes related work on aspect-oriented goal modeling. The paper ends with a conclusion and a discussion of future work in section 6.

2. Background on the Goal-oriented Requirement Language

The *Goal-oriented Requirement Language* (GRL) [4][20][22] is a visual modeling notation that combines the Non-Functional Requirements (NFR) framework [9] and i* framework [24] to support reasoning about goal models. GRL is being standardized by the International Telecommunication Union (ITU) as part of the *User Requirements Notation* (URN) [23]. Within URN, GRL is complemented by *Use Case Maps* (UCMs) [4][7][22], a visual scenario notation for the description of functional requirements and if desired, high-level design. URN is the first standardization effort to address explicitly, in a graphical way, and in one language goals and scenarios, and the links between them. The Eclipse-based jUCMNav tool [20] is the most popular and current URN editing tool, supporting both GRL and UCM modeling.

The syntax of GRL (Fig. 1) is based on the syntax of the i* framework. A GRL goal graph is an AND/OR graph of *intentional elements* that optionally reside within an *actor boundary*. An *actor* represents a stakeholder of the system. A goal graph shows the high-level business goals and non-functional requirements of interest to a stakeholder and the alternatives for achieving these high-level elements. A goal graph also documents rationales (*beliefs*) important to the stakeholder. Intentional elements can be softgoals, goals, tasks, and resources. *Softgoals* differentiate themselves from *goals* in that there is no clear, objective measure of satisfaction for a softgoal whereas a goal is quantifiable. *Tasks* represent solutions to (or *operationalizations* of) goals or softgoals. In order to be achieved or

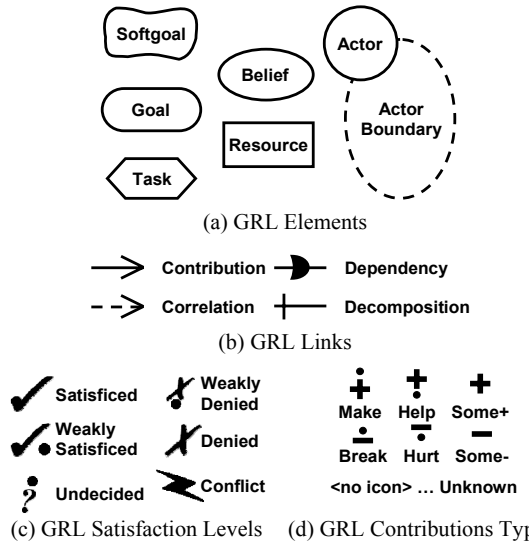

Fig. 1. Summary of jUCMNav's GRL Notation

completed, softgoals, goals, and tasks may require *resources* to be available.

Various kinds of links connect the elements in a goal graph. *Decomposition* links allow an element to be decomposed into sub-elements. AND as well as OR decompositions are supported. *Contribution* links indicate desired impacts of one element on another element. A contribution link has a qualitative contribution type (see Fig. 1.d). *Correlation* links are similar to contribution links, but describe side effects rather than desired impacts. Finally, *dependency* links model relationships between actors (one actor depending on another actor for something).

From the NFR framework, GRL borrows the notion of an evaluation mechanism that supports reasoning about the goal graph. The decisions of a stakeholder are typically documented in the goal graph by the assignment of satisfaction levels (see Fig. 1.c) to alternatives (e.g. the chosen alternative is set to Satisficed whereas all other alternatives are set to Denied). Based on these initial settings and the various links with various contribution types, the satisfaction ratings are propagated to higher-level goals and non-functional requirements of stakeholders. jUCMNav keeps track of these initial settings separate from goal graphs in *strategies*. Several strategies can be defined for a goal model, allowing trade-off analyses to be performed by exploring and comparing various configurations of alternatives. GRL also takes into account that not all high-level goals and non-functional requirements are equally important to the stakeholder. Therefore, jUCMNav supports the definition of *evaluation attributes* for intentional elements such as *priority* (high, medium, low, none), which are also

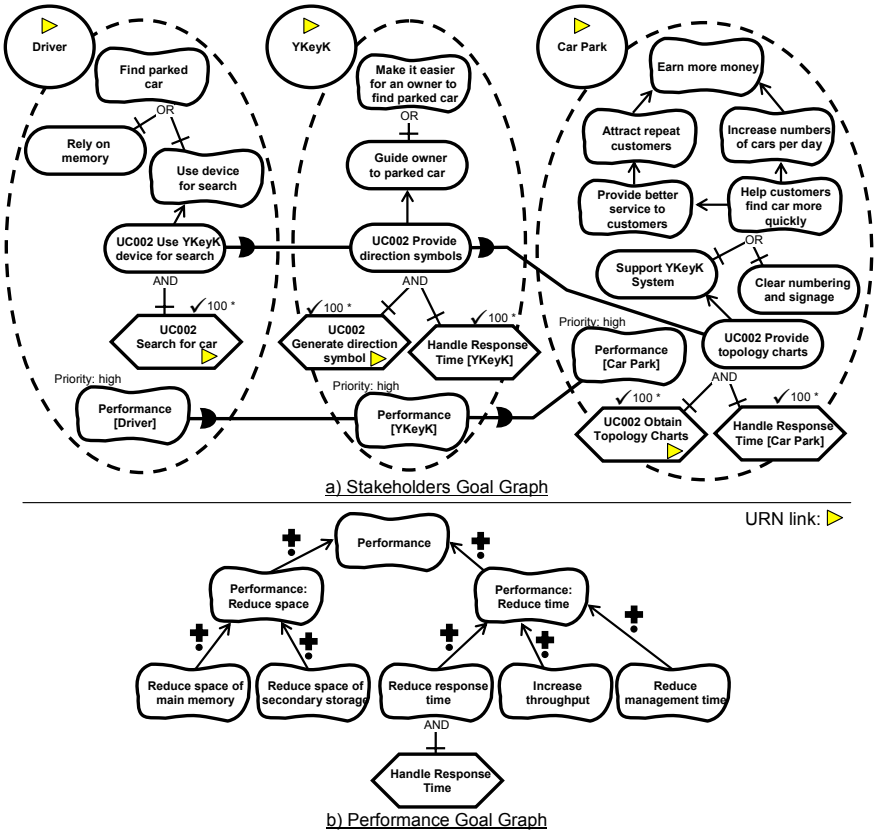

Fig. 2. Simplified GRL Example from YKeyK Case Study

taken into account when evaluating strategies for the goal model [20].

URN allows *URN links*, links between any URN model elements, and jUCMNav uses small triangles to indicate their presence. In particular, links from actors and intentional elements in GRL models to UCM models establish traceability between goal and scenario models in URN (see e.g. the three tasks in Fig. 2.a that reference the UCM model for a more detailed description of use case UC002). As the focus of this paper is on GRL, the UCM models are not further discussed.

The approach we will present here will be illustrated by a simple case study. Fig. 2 shows a simplified version of the YKeyK application. YKeyK stands for Your Key Knows, a system that allows car owners to find their car in a car park by following the directions shown on a small display of the owner's car key. The Stakeholders Goal Graph (Fig. 2.a) illustrates three stakeholders, their main goals, and one alternative. The Driver wants to find a parked car, YKeyK wants to make it easier for the driver to find the car, and the Car Park wants to earn more money. The documented alternative is for the Driver to rely on memory and for the Car Park to provide clear numbering and signage. The goals of the stakeholders are eventually broken down into goals and tasks that describe a possible solution (prefixed with UC002 which stands for Use Case #2). The use case introduces dependencies between the stakeholders as the Driver depends on YKeyK to provide directions in order to search for the car with YKeyK. In turn, YKeyK depends on the Car Park to provide topology charts in order to be able to give correct directions. Furthermore, the Driver is concerned about Performance which introduces further dependencies on YKeyK and the Car Park. In order to address this concern, response time issues have to be handled by YKeyK and the Car Park as indicated by the task Handle Response Time. Note that the impact of the Handle Response Time task on the Performance softgoal is not shown on the Stakeholders Goal Graph, but on the Performance Goal Graph. Finally, initial satisfaction levels and evaluation attributes are also shown in the Stakeholders Goal Graph.

The Performance Goal Graph (Fig. 2.b) is a *GRL catalogue* referenced by the Performance softgoals and Handle Response Time tasks in the Stakeholders Goal Graph. Catalogues are reusable goal models. Unlike most other goal modeling tools, jUCMNav supports at least the saving and loading of catalogues. The tool, however, does not deal well with multiple versions of

the same catalogue in one model as is the case in Fig. 2. The Performance Goal Graph must be duplicated and the copy renamed in order to (a) properly link the Performance in Driver, YKeyK, and Car Park to separate instances of the Performance Goal Graph (as indicated by the [..] notation), and (b) evaluate the goal model for all actors at the same time (e.g. by creating a strategy that assigns Satisficed to Handle Response Time for the Car Park, but only Weakly Satisficed for YKeyK). This is obviously a significant maintenance problem which is further aggravated by the crosscutting nature of the performance concern (i.e. many copies have to be made unless there is another way of specifying the multiple usage of the concern).

A fundamental problem with goal-oriented techniques such as the NFR framework, i*, and GRL is their scalability. Although not as apparent from the small example in Fig. 2, imagine having to consider not just performance, one use case, and three stakeholders with just a few high-level goals and alternatives, but many more non-functional requirements, use cases, stakeholders, high-level goals, and alternatives. Goal graphs quickly become too complicated to manage. It can be argued that it is advantageous for trade-off analyses to see all goals and non-functional requirements at once. We, however, argue that there is a need for a better way of specifying and visualizing complex relationships between intentional elements in a goal graph.

3. Aspect-Oriented GRL

Aspect-oriented GRL (AoGRL) adds support for aspect-oriented modeling to *GRL*. AoGRL combined with Aspect-oriented Use Case Maps (AoUCM, first introduced at REV'06 [14][15][16][22]) form the Aspect-oriented User Requirements Notation (AoURN), unifying goal, scenario, and aspect-oriented modeling in one framework.

AoURN allows requirements engineers to better encapsulate requirements concerns which are hard or impossible to encapsulate with URN models. In the context of URN, the major concerns are use cases and non-functional requirements, all of which can be modeled by aspects with AoURN. In addition, each stakeholder is also modeled as a concern with AoGRL.

3.1. Basic AoGRL Concepts

Generally, an aspect identifies locations in a model (called *joinpoints*) through parameterized expressions (called *pointcuts*). Joinpoints are defined in a *joinpoint model* that clearly defines all possible locations in the modeling or programming language. Furthermore, aspects specify *advice* which will be inserted at the joinpoints. Advice describes properties such as behavior or goals. An aspect is a *crosscutting* concern as the aspect's advice is required at multiple locations in other concerns in the model. The terminology is borrowed from AspectJ [5], but the concepts also apply to other flavors of aspect-oriented programming (AOP) [13] and aspect-oriented modeling (AOM) [8] in general.

3.1.1. Joinpoint model. To fully support aspect-oriented modeling with AoGRL, the aspect concepts mentioned in the previous paragraph have to be introduced to GRL. As a first step, a *joinpoint model* has to be defined for AoGRL. All GRL nodes (i.e. intentional elements and beliefs) optionally residing within the boundary of an actor are deemed to be joinpoints. Therefore, pointcut expressions for AoGRL models can identify any GRL node and advice can be added to any GRL node. With the joinpoint model in place, the following concepts have to be defined for AoGRL: aspect, advice graph, and pointcut graph.

3.1.2. Aspect. An *aspect* is simply an organizational construct that contains all goal graphs required to describe a concern. In particular, it contains (a) any number of advice graphs that specify the crosscutting goal graphs and (b) any number of pointcut graphs that identify the joinpoints where the crosscutting occurs. Both types of goal graphs are structurally the same as traditional GRL goal graphs, allowing the requirements engineer to continue working with familiar models. Just like pointcut and advice maps in AoUCM, pointcut and advice graphs are separate from each other and can therefore be reused separately.

3.1.3. Advice graph and pointcut graph. *Advice graphs* are very similar to the current notion of GRL catalogues if the catalogue describes the goal model of only one concern. AoGRL adds the ability to easily include GRL catalogues multiple times into a GRL model. This is done with the help of *pointcut graphs*, which describe pointcut expressions with partial, parameterized goal graphs. The pointcut expression consists of any specially marked intentional elements or beliefs plus the links among them and the actors in which they reside. A pointcut expression is identified by *pointcut markers* (see the small, dashed diamonds with a P inside in Fig. 3). Parameterization is achieved with the wildcard "*", ensuring that more than one location in the base model can be identified by one single pointcut expression. In addition to "*", logical operators such as "and", "or", and "not" can also be

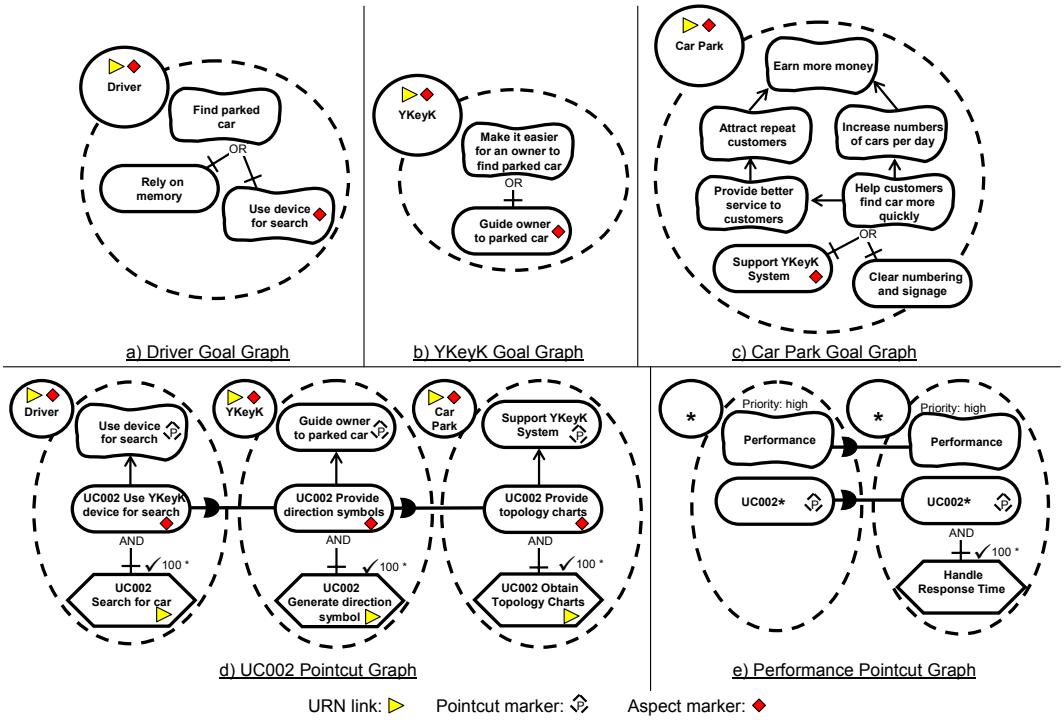

Fig. 3. Simplified AoGRL Example from YKeyK Case Study

used. Note that the ability to mark an element as a pointcut expression is the only extension required for the jUCMNav tool in order to specify AoGRL models. Pointcut expressions are matched against other concerns in the AoGRL model and matches are indicated with small, filled diamonds called *aspect markers* (see stakeholders and some softgoals and goals in Fig. 3).

3.1.4. **Composition rule.** Unlike pointcut maps in AoUCM, pointcut graphs contain not just the pointcut expression, but also other elements, i.e. the elements that are not marked with a pointcut marker. These elements either reference elements with the same name in the aspect's advice graph or are new elements. Connecting these elements with elements of the pointcut expression defines the *composition rule* for the aspect. The composition rule is applied to each joinpoint identified by the pointcut expression (e.g. connecting UC002 Use YKeyK device for search to Use device for search with a contribution link will add such a link and the goal UC002 Use YKeyK device for search and anything else that is connected to the goal on the pointcut graph to the Driver actor when the use case aspect is composed with the goal model). Support for matching and composition will have to be added to the jUCMNav tool. Such support will ensure that aspect markers are added to AoGRL models and that AoGRL models can be navigated with the help of the aspect and pointcut markers.

3.2. YKeyK Case Study

The simplified AoGRL example from Fig. 3 is equivalent to the simplified GRL model from Fig. 2. The AoGRL model contains two aspects, one for use case #2 and one for performance. The first only contains the UC002 Pointcut Graph (Fig. 3.d); the latter contains the Performance Pointcut Graph (Fig. 3.e) and the Performance Advice Graph (which is the same as the Performance Goal Graph from Fig. 2.b). Three more concerns are modeled separately in the AoGRL model: the Driver concern (Fig. 3.a), the YKeyK concern (Fig. 3.b), and the Car Park concern (Fig. 3.c). Note that these three concerns now only model the goals and alternatives related to the stakeholder.

The use case aspect matches the three intentional elements in Fig. 3.a to Fig. 3.c as indicated by the aspect markers, thus adding the use case aspect to the stakeholder concerns. The performance aspect matches the two dependencies in Fig. 3.d, thus adding the Handle Response Time task to the target of the dependency (the goals with the aspect markers in the YKeyK and Car Park actors of Fig. 3.d). In addition, the Performance Pointcut Graph stipulates that Performance goals have to be added to the actors in which the source and target of the dependency reside. Therefore, all actors in Fig. 3.d are also marked with aspect markers, indicating that an aspect adds elements to the ac-

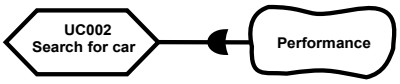

Fig. 4. Aspect Dependencies Goal Graph

tors. Finally, the initial satisfaction levels and evaluation attributes are defined on the pointcut graphs and will be applied to the elements inserted at the identified joinpoints. Note that initial satisfaction levels and evaluation attributes can be defined on advice graphs instead of pointcut graphs, if the levels or attributes are to be applied to all joinpoints matched by the pointcut expressions. In case of a conflict, the pointcut graph specifications override the advice graph specifications. Once all aspects have been applied, the resulting GRL model is the same as the GRL model in Fig. 2.

The pointcut expression in Fig. 3.e matches the use case goals in Fig. 3.d because of a naming convention (prefix UC002 for the goals). Instead of naming conventions, the metadata facility in URN can be used. In this approach, a metadata tag for use case #2 is added to the intentional elements now prefixed with UC002. The pointcut expression is then extended in order to specify metadata tags that need to be matched. More generally, metadata tags could describe classes of elements. We, however, have opted to not pursue this approach at this point, because metadata is not yet visualized on goal graphs. Once visualized, metadata expression will be included in pointcut expressions.

In the YKeyK example, the performance aspect is applied on top of the use case aspect. In this simple example, the order in which aspects need to be applied could be determined by the matching algorithm. In the more general case, however, the matching algorithm requires directions from the requirements engineer. Therefore, a special *Aspect Dependencies* goal graph specifies the order in which aspects need to be applied. Fig. 4 indicates that the performance aspect depends on the use case aspect and therefore needs to be applied after the use case aspect.

4. Evaluation of the YKeyK Case Study

4.1. Models and Metrics

The YKeyK (Your Key Knows) system has been used as a case study for i* models on several occasions [1][2][18]. Based on these i* models, we created three models of the YKeyK system, two GRL models and one AoGRL model.

4.1.1. YKeyK.A. This is a GRL model that contains only one goal graph with the complete system. This model is similar to Fig. 2.a, but without the references to Fig. 2.b and with instances of the referenced goal graph directly copied into Fig. 2.a instead. With a total of almost 300 model elements in one goal graph, the approach for YKeyK.A is clearly not scalable and mostly serves as a base to compare other alternatives.

4.1.2. YKeyK.B. This is the most likely used approach to describe the YKeyK system with GRL. Each stakeholder is described on its own goal graph, referencing if necessary goal graphs for non-functional requirements and use cases. This model is similar to Fig. 2, but splits up Fig. 2.a into separate goal graphs, one for each stakeholder. Only outgoing dependency links are shown on a stakeholder goal graph in order to avoid redundant incoming dependency links (as they are covered by outgoing links on another stakeholder goal graph). Goal graphs for use cases are not shown in Fig. 2 as it is a simplified model. In general, these goal graphs simply show the links between the tasks related to a specific use cases (e.g. the tasks prefixed with UC002 in Fig. 2.a). Furthermore, the YKeyK.B approach assumes that the [..] notation is supported by tools, even though the automatic analysis capabilities of current tools do not support the [..] notation (see the Performance softgoals in Fig. 2.a for examples of the [..] notation).

4.1.3. YKeyK.C. This is an AoGRL model of the YKeyK system. Similar to YKeyK.B, each stakeholder, each non-functional requirement, and each use case is encapsulated by a concern and modeled on separate goal graphs. The stakeholder goal graphs, however, do not reference the goal graphs for non-functional requirements and use cases. Instead, pointcut graphs for each non-functional requirement and use case define how the crosscutting non-functional requirements or use cases are added to the model.

4.1.4. Metrics-based evaluation. The evaluation of the AoURN and URN models is carried out based on metrics for modularity, reusability, understandability, and maintainability of aspect-oriented software available in literature [21]. The metrics are adapted for the use in URN and AoURN models by mapping the notion of component in [21] to *model unit*, the notion of operation to *key model element*, and the notion of line of code to *model element*. Model units are concerns, goal graphs, and UCMs. Key model elements are intentional elements, responsibilities, and stubs. Model elements are any visual URN or AoURN modeling element. As the metrics are merely adapted, the quality model established in [21] with the help of Basili's GQM methodology [6] still applies to this evaluation. The quality model links the metrics to the qualities of

Table 1. Adapted Metrics for URN and AoURN Models

Notion from [21]	URN/AoURN	
Component	Model Unit (MU)	concern, goal graph, UCM
Operation	Key Model Element (KME)	intentional element, responsibility, stub (including references)
LOC	Model Element (ME)	any visual URN/AoURN model element (including references)

A) Separation of concerns metrics		
CDMU (Concern Diffusion over Model Units) counts the number of MUs whose main purpose is to contribute to a concern plus all other MUs that reference them.	**CDKME** (Concern Diffusion over Key Model Elements) counts the number of KMEs whose main purpose is to contribute to a concern.	**CDME** (Concern Diffusion over Model Elements) counts the number of concern switches from a particular concern to any other concern within each MU.
B) Coupling metrics		
CBMU (Coupling between Model Units) is defined for an MU as the number of other MUs it depends on.		
C) Cohesion metrics		
LCOKME (Lack of Cohesion in Key Model Elements) counts the number of KMEs in an MU that do *not* contribute to the main concern of the MU.		
D) Size metrics		
VS (Vocabulary Size) counts the number of MUs in the model.	**NME** (Number of Model Elements) counts the number of MEs in the model.	**AvNME** (Average Number of Model Elements) calculates the average number of MEs per goal graph and UCM in the model.

interest such as reusability, maintainability, and understandability. The mapping and the definition of each metric are summarized in Table 1.

Table 2 shows the results for the metrics applied to the three models with the best results highlighted. In general, the lower the result, the better it is. Not all results, however, make sense, especially considering the pathological situation of YKeyK.A with only one goal graph in the model. Therefore, the artificially low results of the metrics CDMU, CBMU, LCOKME, and VS must not be taken into account.

The AoGRL model of YKeyK.C performs significantly better than the GRL models of YKeyK.A and YKeyK.B except in the following two cases. First, CDKME for YKeyK.A is better because the use case goal graphs in YKeyK.C have to repeat all tasks from the use case pointcut graphs to show the links between these tasks. This does not need to be done for YKeyK.A as there is only a single goal graph. The use case goal graphs, however, are simple and the difference between the results of the CDKME metric is small. Therefore, the results of CDKME do not allow concluding that YKeyK.A is a better approach than YKeyK.C. Second, VS for YKeyK.B is better because the AoGRL-based approach introduces concerns and pointcut graphs to the model, explaining the difference in the results of the VS metric. The concerns, however, are used to group the AoGRL model into more manageable sub-models which each contain a very small number of diagrams. Furthermore, the number of concerns in YKeyK.C (14) is very similar to the vocabulary size of YKeyK.B. Therefore, the results of VS do not allow concluding that YKeyK.B is a better approach than YKeyK.C.

Table 2. Results of Metrics

Metric	*GRL*		*AoGRL*
	YKeyK.A	*YKeyK.B*	*YKeyK.C*
CDMU	n/a (14)	42	21
CDKME	111	154	127
CDME	98	117	0
CBMU	n/a (0)	34	7
LCOKME	n/a (0)	88	0
VS	n/a (1)	13	35
NME	285	320	264
AvNME	285	25	12

4.1.5. Task-based evaluation. In addition to the metrics, the most common update tasks were considered for all three approaches. These tasks cover adding or changing a non-functional requirement, use case, or stakeholder. For YKeyK.A, the tasks always involve finding, in the one large goal graph, all locations that require a change and then changing them (the needle-in-the-haystack approach). For YKeyK.B, the tasks require changing the goal graph for the non-functional requirement or use case and then finding the locations in the stakeholder goal graphs that also require a change and changing them. If a stakeholder is changed, all changes can be made on the stakeholder goal graph, but generally the changes are distributed over many different goal graphs. For YKeyK.C, the tasks require changes to the goal graphs, advice graphs, and pointcut graphs of a single concern (either the one encapsulating the non-functional requirement, the use case, or the stakeholder). In addition, changes to the number of

Table 3. Breakdown of AvNME Metric

Goal Graphs	*YKeyK.B*	*YKeyK.C*
Stakeholders	42	6
NFR / UC	10	10
Pointcuts	n/a	20

matches of pointcut expressions are monitored and reviewed, ensuring that the desired joinpoints are stilled matched.

4.1.6. Summary. Overall, the results suggest that the case study's AoURN model exhibits better *modularity*, *understandability*, *reusability*, and *maintainability* than the URN models. In general, the complexity of stakeholder goal graphs in GRL models is traded against the complexity of pointcut graphs in AoGRL (see Table 3 for a further breakdown of the AvNME metric for YKeyK.B and YKeyK.C). While pointcut expressions can be complex, they allow the complexity to be localized and are therefore more maintainable and understandable without hindering reusability. The results for CDKME and NME are better for YKeyK.C, but somewhat less pronounced. These two results are expected to further improve in larger models with concerns that crosscut more often. The more a concern crosscuts the GRL model, the more AoGRL can reduce the complexity of goal graphs without necessarily requiring increasingly complex pointcut expressions. For *scalability*, we argue that the reduced complexity of goal graphs (see AvNME results), the ability to group goal graphs with concerns, the encapsulation provided by concerns, the ability to use parameterized pointcut expressions in AoGRL, and the simpler update tasks for AoGRL suggest that AoGRL models are more scalable than GRL models.

4.2. Observations

When modeling the YKeyK system with AoGRL based on the i* models, we made the following general observations. After iteratively modeling all non-functional requirements and all use cases as separate concerns in the AoGRL model, no stakeholders contained any intentional element that belonged predominantly to the stakeholder. The original i* models actually did not model any major goals or alternatives for any of the stakeholders. This presents anecdotal evidence that modeling non-functional requirements *and* use cases as separate concerns in goal-oriented models allows the requirements engineer to focus on the all-important goals and alternatives of the key stakeholders.

i* models generally consist of two different types of models. The *Strategic Dependency Model* (SDM) shows which actor depends on which other actor and why. The *Strategic Rationale Model* (SRM) is a refinement of the SDM which focuses on one actor at the time, linking the dependencies from the SDM to intentional elements inside the actor. This case study did not define an SDM because the SDM can be derived from the SRMs of all actors regardless of the chosen modeling approach.

The last observation relates to the need to see a composed system much like the model in YKeyK.A where all interactions are visible at once. While the YKeyK.A approach is impractical, we found that the AoGRL model alone is also not sufficient. Some form of composed system needs to be made available to the requirements engineer. We propose to use several views to help with this problem. The views are connected to the aspect and pointcut markers in the AoGRL model. Note that the views are not yet supported by the jUCMNav tool.

Selecting a pointcut marker gives a list of all joinpoints that matched the whole pointcut expression. For each match, the list includes the name of the matched joinpoint and its actor (see Fig. 5.a for the list of the pointcut marker of goal UC002* in Fig. 3.e). When a match is selected, the tool displays the goal graph with the matched joinpoints.

	Joinpoints matched against pointcut expression of *Performance Pointcut Graph*:	
	.UC002	*.UC002*
1	Driver.UC002 Use YKeyK device for search	YKeyK.UC002 Provide direction symbols
2	YKeyK.UC002 Provide direction symbols	Car Park.UC002 Provide topology charts

a) List for Pointcut Marker

	Aspects that matched *Driver.Use device for search*
1	Driver.Use device for search Contribution link from Driver.UC002 Use YKeyK for search
	Pointcut Graph: UC002 Pointcut Graph

b) List for Aspect Marker of Intentional Element (Single Element View)

	Aspects that matched *Driver.Use device for search*		
1	Driver.Use device for search Contribution link from Driver.UC002 Use YKeyK for search	YKeyK.Guide owner to parked car Contribution link from YKeyK.UC002 Provide direction symbols	Car Park.Support YKeyK System Contribution link from Car Park.UC002 Provide topology charts
	Pointcut Graph: UC002 Pointcut Graph		

c) List for Aspect Marker of Intentional Element (Full Match View)

Fig. 5. Views for Pointcut and Aspect Marker

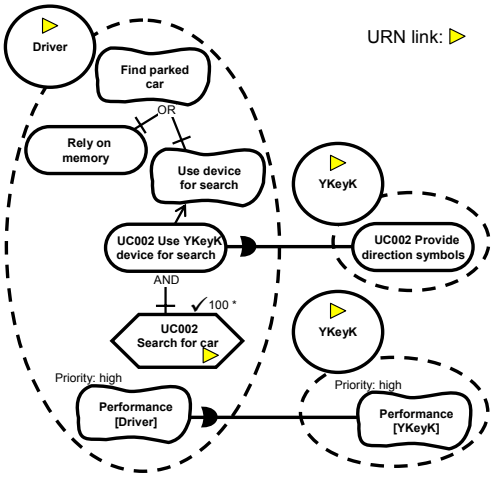

Fig. 6. Composed Goal Graph for Driver

Selecting an aspect marker located inside an intentional element gives a list of all pointcut expressions that matched the intentional element. For each match, the list includes the name of the pointcut graph, the type of the added link, and the node attached to the link. There are two options available for the aspect marker view. In the full match view, the match of the whole pointcut expression is shown (see Fig. 5.c for the full view of the aspect marker of Use device for search in Fig. 3.a). In the single element view, only the selected element is shown (see Fig. 3.b). When a match is selected in any view, the tool displays the pointcut graph which matched the intentional element.

Selecting an aspect marker of an actor displays the composed goal graph of the actor with all additions of all aspects (see Fig. 6 for the composed goal graph of the Driver). Note that the composed goal graph for the Driver in YKeyK.C is the same as the manually created Driver goal graph in YKeyK.B. Also note that a good auto-layout mechanism is important for this view. Even though the auto-layout mechanism of jUCMNav provides an initial layout of the composed goal graph, there is still room for further improvement. While first results indicate that these views are useful, further evaluations are required in general for all views proposed in this section.

5. Related Work on Aspect-Oriented Goal Modeling

Yu *et al.* [25] identify aspects in goal models based on relationships between functional and non-functional goals, propose goal aspects as a way to address scalability issues in goal models, and point out that the syntax of goal aspects requires further research. An example shows a textual definition of a goal aspect. AoGRL uses a visual syntax for aspect-oriented goal models including pointcut expressions.

Alencar *et al.* [1][2][3] identify aspects in i* models and extend the notation to represent aspect-oriented concepts. The extensions, however, do not allow crosscutting elements to be fully separated from other concerns, and pointcut expressions are again defined in a textual way. AoGRL fully encapsulates crosscutting concerns in an aspect. Furthermore, Alencar *et al.* do not identify scenarios as aspects or concerns in i* models whereas AoGRL does model scenarios as concerns because AoGRL is part of the larger AoURN context.

Niu and Easterbrook [17] focus mostly on discovering aspects in goal models with the Repertory Grid Technique and less on how to model discovered aspects.

Kaiya and Saeki [12] propose a pattern-based technique to compose viewpoints. Like AoURN, [12] investigates goal and scenario models at the same time, but limits the composition technique to a simple combinatorial approach instead of more powerful pointcut expressions as in AoURN. Furthermore, it is not clear whether the approach specifies all required information for the composition of goal graphs and use cases.

Finally to the best of our knowledge, a substantial, quantitative analysis of an aspect-oriented and non-aspect-oriented goal model for the same system has not been attempted before for i*, NFR, or GRL models.

6. Conclusion and Future Work

This paper presents Aspect-oriented GRL (AoGRL) as a way to manage the complexity of goal models such as GRL, i*, and the NFR Framework. AoGRL is part of AoURN, a unified framework for modeling crosscutting concerns in goal models and scenario models in an aspect-oriented way. Crosscutting concerns in AoURN are use cases and non-functional requirements of any kind. AoGRL visually models aspects including parameterized pointcut expressions without the need for new notational concepts. Reusable GRL models (catalogues) can now more easily be included multiple times in a goal model. The complexity of goal graphs in GRL is reduced by restructuring elements related to crosscutting concerns, allowing for a better way of specifying complex relationships between intentional elements. AoGRL models not only non-functional requirements as concerns, but also use cases and stakeholders, allowing requirements engineers to more easily focus on the major goals and alternatives of key stakeholders.

The YKeyK case study compares two GRL models and one AoGRL model of the same system. The

evaluation is a first attempt to obtain more quantifiable results. More controlled experiments with varied, real-world sized case studies are necessary to better understand and validate the adapted metrics for separation of concerns, coupling, cohesion, and size [21]. Despite its limitations, the evaluation establishes groundwork for further investigations into quantifiable measures for AoURN and URN models and suggests that AoURN models improve modularity, understandability, maintainability, reusability, and scalability of URN models.

Besides further evaluations, future work will focus on implementing modeling support for AoGRL and the proposed views in the jUCMNav tool as well as extending the existing evaluation mechanism of jUCMNav to AoGRL. How best to link, combine and evolve goals and scenarios in single aspects also requires further research. Furthermore, aspect-oriented modeling of GRL strategies and UCM scenarios defined as a specific trace through a UCM model where only one alternative is taken at a choice point has to be examined in more detail.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada, through its programs of Discovery Grants and Postgraduate Scholarships, and by the Ontario Research Network on e-Commerce.

References

[1] Alencar, F., Castro, J., Moreira, A., Araújo, J, Silva, C., Monteiro, C., Ramos, R., and Mylopoulos, J.: Simpifying i* Models. *17th Intl. Workshop on Agent-Oriented Information Systems (AOIS-2007)*, Trondheim, Norway (June 2007)

[2] Alencar, F., Moreira, A., Araújo, J, Castro, J., Ramos, R., and Silva, C.: Proposal to deal with the Complexity of i* Models with Aspects. *1st Intl. Conf. on Research Challenges in Info. Science (RCIS'07)*, Quarzazate, Marocco (April'07)

[3] Alencar, F., Moreira, A., Araújo, J, Castro, J., Silva, C., and Mylopoulos, J.: Towards an Approach to Integrate i* with Aspects. *8th Intl. Bi-Conf. Workshop on Agent-Oriented Information Systems (AOIS-2006)*, Luxembourg (June 2006)

[4] Amyot, D.: Introduction to the User Requirements Notation: Learning by Example. *Computer Networks*, Vol. 42(3), pp 285-301 (June 21, 2003)

[5] *AspectJ web site*. http://www.eclipse.org/aspectj/ (accessed July 2007)

[6] Basili, V., Caldiera, G., and Rombach, H.: The Goal Question Metric Approach. *Encyclopedia of Soft. Eng.*, Vol. 2, pp. 528-532, John Wiley & Sons, Inc. (September 1994)

[7] Buhr, R.J.A. and Casselman, R.S.: *Use Case Maps for Object-Oriented Systems*. Prentice-Hall (1995)

[8] Chitchyan, R. *et al.*: *Survey of Analysis and Design Approaches*. AOSD-Europe Report ULANC-9 (May 2005), http://www.aosd-europe.net/deliverables/d11.pdf (accessed July 2007)

[9] Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J.: *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, Dordrecht, USA, 2000.

[10] *Early Aspects website*. http://www.early-aspects.net/ (accessed July 2007)

[11] Jacobson, I. and Ng, P.-W.: *Aspect-Oriented Software Development with Use Cases*. Addison-Wesley (2005)

[12] Kaiya, H. and Saeki, M.: Weaving Multiple Viewpoint Specifications in Goal-Oriented Requirements Analysis. *11th Asia-Pacific Softw. Eng. Conf. (APSEC 2004)*, Busan, Korea, IEEE Computer Society Press, pp 418-427 (Nov. 2004)

[13] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., and Irwin, J.: Aspect-Oriented Programming. *ECOOP'97–Object Oriented Progr., 11th Europ. Conf.*, LNCS 1241, Springer, pp 220-242 (June 1997)

[14] Mussbacher, G., Amyot, D., and Weiss M.: Visualizing Aspect-Oriented Requirements Scenarios with Use Case Maps. *International Workshop on Requirements Engineering Visualization (REV 2006)*, Minneapolis, USA (Sep. 2006)

[15] Mussbacher, G., Amyot, D., and Weiss M.: Visualizing Early Aspects with Use Case Maps. To appear in: *Transactions on Aspect-Oriented Software Development*.

[16] Mussbacher, G., Amyot, D., Whittle, J., and Weiss M.: Flexible and Expressive Composition Rules with Aspect-oriented Use Case Maps (AoUCM). *10th Intl. Workshop on Early Aspects (EA 2007)*, Vancouver, Canada (March 2007)

[17] Niu, N. and Easterbrook S.: Discovering Aspects in Requirements with Repertory Grid. *Wksh. on Early Aspects*, co-located with ICSE 2006, Shanghai, China (May 2006)

[18] Rashid, A. Moreira, A., and Araújo, J.: Modularisation and Composition of Aspectual Requirements. *2nd Int. Conf. on Aspect-Oriented Software Development (AOSD 2003)*, Boston, USA, ACM Press, pp 11-20 (March, 2003)

[19] Rashid, A., Moreira, A., Araújo, J, and Chitchyan, R.: Aspect-Oriented Requirements Engineering. Tutorial at *14th International Requirements Engineering Conference (RE 2006)*. Minneapolis, USA (September 2007)

[20] Roy, J-F.: *Requirements Engineering with URN: Integrating Goals and Scenarios*. M.Sc. thesis, OCICS, University of Ottawa, Canada (2007), (accessed July 2007) http://www.softwareengineering.ca/jucmnav

[21] Sant'Anna, C. *et al.*: On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework. *Brazilian Symposium on Software Engineering (SBES'03)*, Manaus, Brazil, pp 19-34 (October 2003)

[22] UCM Virtual Library. (accessed July 2007) http://www.UseCaseMaps.org/pub/

[23] *User Requirements Notation (URN) – Language Requirements and Framework*, ITU-T Recommendation Z.150. Geneva, Switzerland (February 2003), (accessed July 2007) http://www.itu.int/ITU-T/publications/recs.html

[24] Yu, E.: *Modeling Strategic Relationships for Process Reengineering*. Ph.D. thesis, Department of Computer Science, University of Toronto, Canada (1995)

[25] Yu, Y., Leite, J. C. S. d. P., and Mylopoulos, J.: From Goals to Aspects: Discovering Aspects from Requirements Goal Models. *12th International Requirements Engineering Conference (RE'04)*, Kyoto, Japan (September 2004)