

An Automated Change Impact Analysis Approach to GRL Models

Hasan Salim Alkaf¹, Jameleddine Hassine¹, and Abdelwahab Hamou-Lhadj²
and Luay Alawneh³

¹ Department of Information and Computer Science
King Fahd University of Petroleum and Minerals, Dahrán, Saudi Arabia
g201201840@kfupm.edu.sa, jhassine@kfupm.edu.sa

² Electrical and Computer Engineering Department
Concordia University, Montréal, Canada
abdelw@ece.concordia.ca

³ Department of Software Engineering
Jordan University of Science and Technology, Irbid, Jordan
lmalawneh@just.edu.jo

Abstract. Goal-oriented approaches to requirements engineering have gained momentum with the development of many frameworks, methods, and tools. As stakeholders' needs evolve, goal models evolve quickly and undergo many changes in order to accommodate the rapid changes of stakeholders' goals, technologies, and business environments. Therefore, there is a need for mechanisms to identify and analyze the impact of changes in goal models. In this paper, we propose a Change Impact Analysis (CIA) approach to Goal-oriented Requirements Language (GRL), part of ITU-T's User Requirement Notation (URN) standard. Given a suggested modification within a given GRL model, our approach allows for the identification of all impacted GRL elements within the targeted model as well as across all GRL models that are linked to it through URN links. Furthermore, the proposed approach allows for the identification of the potentially impacted GRL evaluation strategies. The developed GRL-based CIA approach is implemented as a feature within the Eclipse-based jUCMNav framework. We demonstrate the applicability of our approach using two real-world GRL specifications.

1 Introduction

Goal-oriented requirements engineering (GORE) is concerned with helping stakeholders understand, elaborate, analyze, and document their requirements. Goal modeling is becoming a popular way for describing and connecting stakeholders' intentions and goals with technical requirements. Goals are used to capture, at different levels of abstraction (ranging from high-level strategic mission statements to low-level operational tasks), the various objectives the system under development should accomplish or the concerns that stakeholders may have with it. The growing popularity of goal-oriented modeling, and its adoption by a large international community, led to the development of many goal-oriented modeling

languages and notations, e.g., i^* [1], TROPOS [2], and the Goal-oriented Requirements Language (GRL) [3], part of ITU-T’s User Requirements Notation (URN) standard.


Although, goals are supposed to be more stable than the requirements that helped model them [4], due to continuous changes in the business environment and to the sustained technological advances, goal models are deemed to change accordingly. Commonly, when a change is made, there is often a ripple effect through the goal model. Hence, there is a need to trace such ripple effects across the goal model and identify the potential consequences of such impact on stakeholders’ goals. Change Impact Analysis (CIA) is defined by Bohner and Arnold [5] as ”identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change”. Although change impact analysis techniques have been mostly used at lower levels of abstractions (e.g., code level [6]), many techniques have been developed to target other software artifacts, such as architectural models, software specifications, data sources, configuration files, etc.






The main motivation of this research is to apply change impact analysis to goal-oriented models. In particular, we are interested in understanding and capturing how changes propagate through GRL models. In this paper, we extend and build upon our preliminary work [7]. The paper serves the following purposes:





- It provides a GRL-based approach to Change Impact Analysis (CIA). The proposed CIA approach allows maintainers and analysts to understand how a change in a GRL model is propagated within the model itself (e.g., between actors of the model) and across other GRL models (i.e., GRL to GRL propagation) through URN links. Furthermore, the proposed approach allows for the identification of the potentially impacted GRL evaluation strategies as a result of a proposed change.
- It provides a prototype tool that automates the proposed GRL-based change impact analysis approach. The prototype is implemented as a feature within the jUCMNav [8] tool and is publicly available.
- It demonstrates the applicability of our approach and tests our prototype tool, using two real-world GRL specifications presenting different constructs and features, namely, Adverse Event Management System (AEMS) and a commuting system.



The rest of this paper is organized as follows. Section 2 provides a brief overview of the Goal-Oriented Language (GRL). In Sect. 3, we present our proposed GRL-based Change Impact Analysis (CIA) approach along with the prototype tool. The applicability of the proposed approach is demonstrated in Sect. 4. A discussion of the related work, the benefits and limitations of our approach is provided in Sect. 5. Finally, conclusions and future work are presented in Sect. 6.

2 GRL in a nutshell

The Goal-oriented Requirement Language (GRL) [3], part of ITU-T's User Requirement Notation (URN) standard, is a visual modeling notation that is used to model intentions, business goals, functional and non-functional requirements (NFR). A GRL goal model is a graph of intentional elements, that optionally reside within an actor. Actors (illustrated as ) are holders of intentions; they are the active entities in the system or its environment who want goals to be achieved, tasks to be performed, resources to be available, and softgoals to be satisfied [3]. Actor definitions are often used to represent stakeholders as well as systems. A GRL actor may contain intentional elements and indicators describing its intentions, capabilities and related measures.

Softgoals (illustrated as ) differentiate themselves from goals (illustrated as ) in that there is no clear, objective measure of satisfaction for a softgoal whereas a goal is quantifiable, often in a binary way. Tasks (illustrated as ) represent solutions to (or operationalizations of) goals or softgoals. In order to be achieved or completed, softgoals, goals, and tasks may require resources (illustrated as ) to be available. A GRL indicator (illustrated as ) is a GRL element that is used to represent some real-world measurements. An indicator usually convert real-world values in user-defined units into GRL satisfaction values on a standard scale (e.g. [-100, 100]).

Various kinds of links connect the elements in a goal graph. Decomposition links (illustrated as ) allow an element to be decomposed into sub-elements (using AND, OR, or XOR). Contribution links (illustrated as ) indicate desired impacts of one element on another element. A contribution link has a qualitative contribution type (e.g., Make, Help, SomePositive, Unknown, SomeNegative, Break, Hurt) and/or a quantitative contribution (e.g., an integer value within [-100, 100]). Correlation links (illustrated as ) describe side effects rather than desired impacts. Dependency links (illustrated as ) model relationships between actors, where intentional elements inside actor definitions can be used as source and/or destination of a dependency link. In this research, we adopt the classification of GRL dependencies introduced in [9] that considers contributions, correlations and decompositions links as implicit dependencies, and dependency links as explicit dependencies.

Initial satisfaction levels, which can be quantitative (e.g., within [-100, 100]), or qualitative (e.g., Satisfied, Weakly Satisfied, Denied, Weakly Denied, etc.) of some of the intentional elements constitute a GRL strategy. These initial values (emanating from a contextual or a future situation) propagate to the other intentional elements of the model through the various model links, allowing for the assessment of how high-level goals are achieved and may reveal more appropriate alternative strategies. Finally, URN links (illustrated as a black triangle symbol  (source)  (target)) are used to connect a source URN model element with a target URN model element. URN links model user-defined relationships such as traceability, refinement, implementation, etc. For a detailed description of the GRL language, the reader is invited to consult [3].

3 GRL Change Impact Analysis (CIA) Approach

Figure 1 describes the proposed GRL-based change impact analysis approach. To identify the impact of a change in a GRL model under maintenance, an analyst may select a GRL construct (i.e., an intentional element, an indicator, or a link) to be changed, then specify the type of change (e.g., addition, modification, deletion). Next, the GRL Model Dependency Graph (GMDG) is constructed (see Sect. 3.1), then sliced according to the specified slicing criterion (see Sect. 3.2). GMDG impacted nodes are then identified, mapped back to the original GRL model, and marked with a different color. Finally, impacted evaluation strategies and impacted URN links are displayed as a GRL comment construct (see Sect. 3.5).

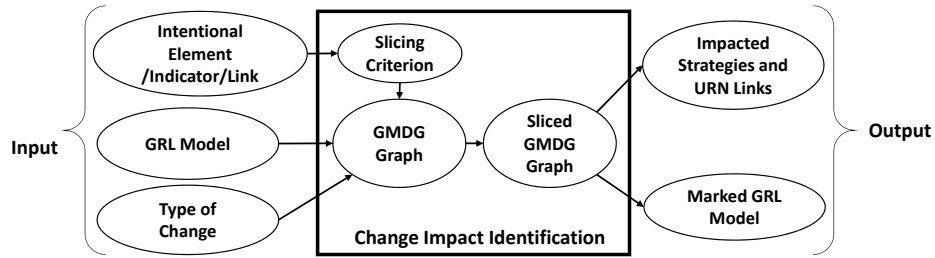


Fig. 1. GRL CIA Approach

In what follows, we provide some necessary definitions (adopted and modified from [7]) that are used in the subsequent sections.

Definition 1 (GRL Model). We assume that a GRL model GRLM is denoted by a 3-tuple: (Actors, Elements, Links), where:

- Actors is the set of actor references in the GRL model.
- Elements is the set of intentional elements (i.e., tasks, goals, softgoals, resources) and indicators in the GRL model.
- Links is the set of links in the GRL model.

It is worth noting that we don't consider collapsed actors (although they are described in the URN standard [3]), since they are not supported in jUCM-Nav [8].

Definition 2 (GRL Link). We define a GRL link as $(type, src, dest)$: $LinkTypes \times Elements \times Elements$, where $LinkTypes = \{contribution, correlation, dependency, decomposition\}$, src and $dest$ are the source and destination of the link, respectively.

Definition 3 (GRL Link Access Functions). Let $l=(type, src, dest)$ be a GRL link. We define the following access functions over GRL links:

- **TypeLink**: $\text{Links} \rightarrow \text{LinkTypes}$, returns the link type (i.e., $\text{TypeLink}(l) = \text{type}$).
- **Source**: $\text{Links} \rightarrow \text{Elements}$, returns the intentional element source of the link (i.e., $\text{Source}(l) = \text{src}$).
- **Destination**: $\text{Links} \rightarrow \text{Elements}$, returns the intentional element destination of the link (i.e., $\text{Destination}(l) = \text{dest}$).

3.1 GRL Model Dependency Graph (GMDG)

In this section, we define the GMDG graph and present the algorithm (Alg. 1) to construct it.

Definition 4 (GRL Model Dependency Graph (GMDG)). A GRL Model Dependency Graph (GMDG) is defined as a directed graph $\text{GMDG}=(N, E)$, where:

- N is a set of nodes. Each GRL intentional element, indicator, or a link is mapped to a node $n \in N$.
- E is a set of directed edges. An edge $e \in E$ represents a dependency between 2 nodes in GMDG and it is illustrated as a solid arrow (\longrightarrow).

First, for each intentional element, indicator, or a link a new GMDG node is created. Next, depending on the type of the GRL links, GMDG dependency links are created between GMDG nodes (i.e., $\text{CreateDependencyLinkGMDG}(e1, e2)$ creates a GMDG dependency link from $e1$ to $e2$).

Algorithm 1: Constructing a GRL Model Dependency Graph (GMDG)

Procedure Name: ConstructGMDG

Input : A GRL Model: (Actors, Elements, Links)

Output: A GRL Model Dependency Graph (GMDG)

foreach $e \in \text{Elements}$ **do**

$n = \text{createGMDGNode}(e)$;

end

foreach $e \in \text{Links}$ **do**

$n = \text{createGMDGNode}(e)$;

if ($\text{TypeLink}(e) == \text{contribution}$ or $\text{TypeLink}(e) == \text{correlation}$ or $\text{TypeLink}(e) == \text{decomposition}$) **then**

$\text{CreateDependencyLinkGMDG}(\text{Destination}(e), \text{Source}(e))$;

$\text{CreateDependencyLinkGMDG}(\text{Destination}(e), n)$;

else

$\triangleright \text{TypeLink}(e) == \text{Dependency}$

$\text{CreateDependencyLinkGMDG}(\text{Source}(e), \text{Destination}(e))$;

$\text{CreateDependencyLinkGMDG}(\text{Source}(e), n)$;

end

end

Figure 2 illustrates a generic GRL model along with its corresponding GMDG graph. Each goal/contribution/decomposition/dependency is represented as a GMDG node. The satisfaction of $G2$ depends on the satisfaction of $G5$ and the contribution type (*help* in this case), hence, two GMDG links are created: (1) between $G2$ and $G5$ and (2) between $G2$ and $Contrib-G5G2$. Since $G1$ is decomposed into $G3$ and $G4$ (using AND-decomposition), four GMDG dependency links are created: (1) one between $G1$ and $G3$, (2) one between $G1$ and $G4$, (3) one between $G1$ and $AND-Decomp-G3G1$, and (4) one between $G1$ and $AND-Decomp-G4G1$. Finally, $G1$ depends on $G2$, which is mapped as two GMDG links: (1) one between $G1$ and $G2$, and (2) one between $G1$ and $depend-G1G2$.

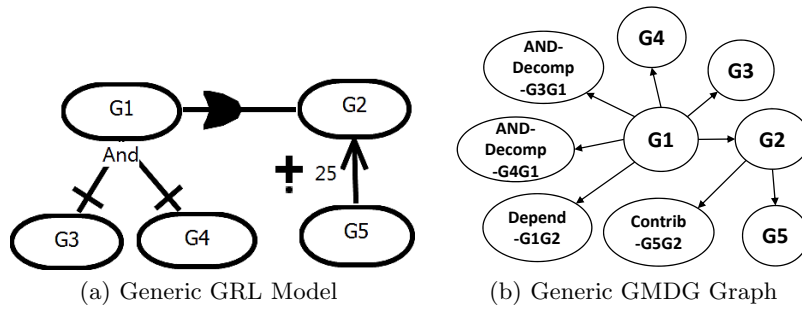


Fig. 2. A Generic GRL model and its corresponding GMDG

3.2 Slicing the GRL Model Dependency Graph

Program Slicing, introduced by Weiser [10] in the early 1980's, is a reduction technique used to decrease the size of a program source code by keeping only the lines within a program that are related to the execution of a specific slicing criterion specified by the user. In order to perform a change impact analysis on GRL models, we extend the concept of program slicing to GMDG graphs. In what follows, we introduce the notion of GRL slicing criterion, then we present the GMDG slicing algorithm (see Alg. 2).

Definition 5 (GRL Slicing Criterion). *Let GRLM be a GRL model. A slicing criterion SC for GRLM may be either a GRL intentional element/Indicators or a GRL link.*

The slicing of the GMDG (see Algorithm 2) is based on a backward traversal of the GMDG. It requires as input the GMDG graph and the GMDG node that corresponds to the slicing criterion SC. The algorithm starts by adding the GMDG node (called *ImpactedGMDGNode*) to the set of impacted nodes (i.e., *SetGMDGImpactedNodes*). Next, it follows each incoming link leading to

ImpactedGMDGNode and add its source to *SetGMDGImpactedNodes*. Finally, a recursive call is made by passing the GMDG and the new reached GMDG node.

Algorithm 2: GMDG Backward Slicing Algorithm

Function Name: SlicingGMDG
Input : A GMDG + GMDG node corresponding to SC
 (LocationInGMDG(SC))
Output: SetGMDGImpactedNodes
 ImpactedGMDGNode = LocationInGMDG(SC);
if *ImpactedGMDGNode* \notin *SetGMDGImpactedNodes* **then**
 | AddToImpactedNodes(ImpactedGMDGNode, SetGMDGImpactedNodes);
 | **if** *hasIncomingLinks*(*ImpactedGMDGNode*) **then**
 | | **foreach** *incomingLink* **do**
 | | | AddToImpactedNodes(Source(*incomingLink*),
 | | | SetGMDGImpactedNodes);
 | | | GMDGslicingAlg(GMDG, ImpactedGMDGNode);
 | | **end**
 | **end**
end

The resulting set of impacted GMDG nodes (i.e., *SetGMDGImpactedNodes*) is then mapped back to *SetGRLImpactedElements*, the set of the original GRL model elements. The elements within *SetGRLImpactedElements*, along with the impacted elements emanating from following the URN links (see Sect. 3.3), are then marked in purple color (see examples in Sect. 4).

3.3 Impact Through URN Links

This step aims at identifying other potential GRL impacted elements by following existing URN links. A URN link is used to create a connection between any two URN elements, e.g., intentional element reference/definition, actor reference/definition, link, etc. A URN link may be defined as follows:

Definition 6 (URN Links). *A URN link is defined as $urnl = (type, from, to)$, where (1) type denotes a user-defined URN link type, (2) from denotes the ID of source URN element, and (3) to denotes the ID of the target URN element.*

Algorithm 3 iterates through the set of impacted elements (i.e., *SetGRLImpactedElements*) and checks whether these elements are involved in any URN link, as source (i.e., *from* field) or as a target (i.e., *to* field). Since an impacted element can serve as a source or a target in a URN link and since one source element can be linked to many target elements and vice versa, we have used two search functions to retrieve the set of elements IDs depending whether we are looking for source or target IDs. (i.e., *searchSourceURNLinks* and *searchTargetURNLinks*). The new identified elements are then add to the set *SetGRLImpactedElements*.

Algorithm 3: Excerpt of the algorithm to identify impacted elements emanating from URN links

Function Name: IdentificationOfOverallImpactedElements
Input : GRL Model + SetGRLImpactedElements
Output: SetGRLImpactedElements
URNLinksList = getAllURNLinks();
foreach $e \in \text{SetGRLImpactedElements}$ **do**
 ▷ Search for target elements IDs when e is defined as source;
 ToElementList = searchTargetURNLinks(e,from,URNLinksList);
 AddToGRLImpactedElements(ToElement, SetGRLImpactedElements);
 ▷ Search for source elements IDs when e is defined as target
 FromElementList = searchSourceURNLinks(e, URNLinksList);
 AddToGRLImpactedElements(FromElement, SetGRLImpactedElements);
end

3.4 Identification of the Impacted GRL Strategies

Once the set of impacted GRL model elements (i.e., *SetGRLImpactedElements*) is identified, we have to spot all impacted evaluation strategies. Algorithm 4 accepts as input a GRL model and the set of impacted GRL elements (*SetGRLImpactedElements* resulting from applying the GMDG slicing algorithm), and produces the set of impacted GRL strategies (i.e., *SetImpactedStrategies*).

Algorithm 4: Identification of the impacted GRL evaluation strategies

Function Name: IdentificationOfImpactedStrategies
Input : GRL Model + SetGRLImpactedElements
Output: SetImpactedStrategies
SetImpactedStrategies = \emptyset ;
StrategiesList = getAllStrategies();
foreach $strategy \in \text{StrategiesList}$ **do**
 foreach $impactedElement \in \text{SetGRLImpactedElements}$ **do**
 if *PartOfStrategy(impactedElement, strategy)* **then**
 AddToImpactedStrategies(strategy, SetImpactedStrategies) ;
 end
 end
end

3.5 jUCMNav GRL-based Change Impact Analysis Feature

Our proposed change impact analysis approach is implemented as a feature ¹ within the jUCMNav framework [8], a full graphical editor and analysis tool for GRL models developed as an Eclipse-based plug-in.

¹ The CIA feature is publicly available and can be downloaded from <https://github.com/JUCMNAV/projetseg/tree/grl>.

To exercise this feature, the user starts by selecting a GRL intentional element, an indicator or a link, then right-clicks to choose from three sub-menu commands: *Addition*, *Deletion*, or *Modification* (see Fig. 3). For the addition option, it is required that the analyst adds the GRL construct first then call the feature. The deletion is provided as a separate option because there will be impacted elements due to the loss of connectivity caused by the deletion. It is worth noting that this CIA menu is activated for the supported GRL constructs only.

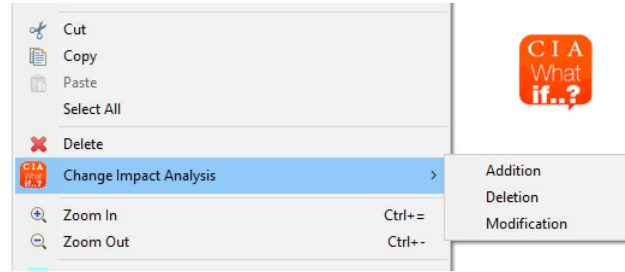


Fig. 3. GRL CIA included in command menu of jUCMNav framework

If any of the impacted element (marked in purple color (see Fig. 6)), is part of a GRL evaluation strategy, the details of the impacted element will appear as a GRL comment (in gray color) with its name, ID, and the name of strategies it belongs to (see Fig. 8(a)). Similarly, information about impacted URN links, such as SourceID, TargetID, and Type, are also shown in the same GRL comment box (see Fig. 7).

4 Empirical Evaluation

In this section, we evaluate our proposed GRL change impact analysis approach using two real-world GRL case studies of different sizes, complexity, and features. Table 1 provides some characteristics of the used case studies.

GRL Spec.	Nb. of GRL Models	Nb. of Intentional Elements	Nb. of Links	Nb. of URN Links	Nb. of Actors
AEMS	5	30	27	6	9
Commuting System	4	19	37	10	3

Table 1. Case studies characteristics

4.1 Case Study 1: Adverse Event Management System (AEMS)

This case study describes an adverse event management system (AEMS) for a hospital. Figure 4 illustrates one of the five GRL models constituting the case study.

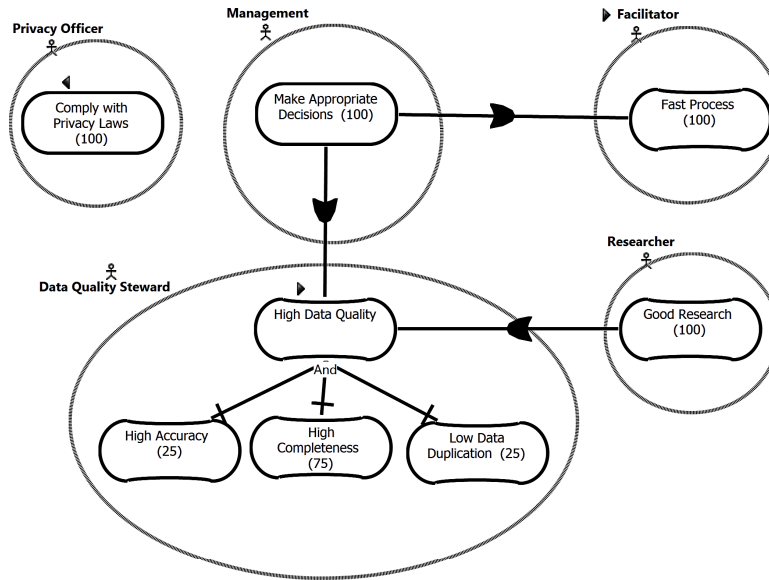


Fig. 4. AEMS GRL Model

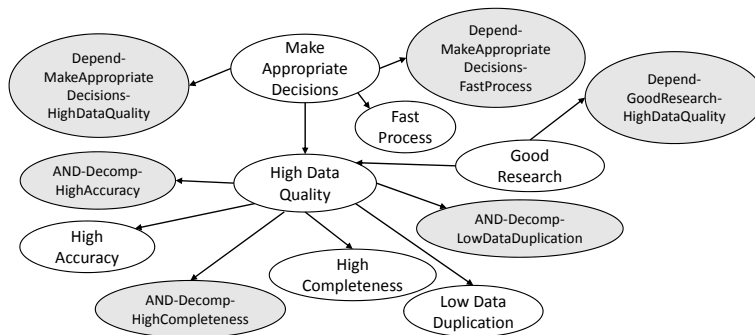


Fig. 5. GMDG Graph corresponding to the AEMS GRL model of Fig. 4

The first CIA task aims to identify potential impacted elements if we modify softgoal *FastProcess* (i.e., the GMDG node corresponding to *FastProcess* is used as slicing criterion to execute Algorithm 2). The produced GMDG is shown in Fig. 5, while the impacted GRL elements are shown in Fig. 6. Since the goal *comply with Privacy Laws* is only linked to the rest of the model through a URN link, called *trace* (having its source at softgoal *High Data Quality*), there is no GMDG node associated with it.

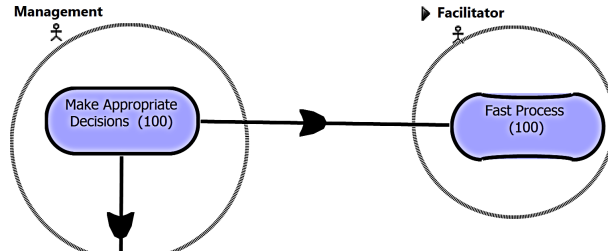


Fig. 6. Impacted elements of the first AEMS CIA task

The second CIA task aims to identify potential impacted elements once we modify the softgoal *High Data Quality*. Three elements are impacted (i.e., goal *Make Appropriate Decisions*, and softgoals *High Data Quality* and *Good Research*) as a result of slicing the GMDG graph with the GMDG node that corresponds to *High Data Quality* as slicing criterion. In addition, goal *Comply with Privacy Law* is impacted since it is the target of the URN link *trace*, having its source at softgoal *High Data Quality*. Finally, one evaluation strategy is identified, called *AsIsAnalysis-Summer2010*, involving both softgoals *High Data Quality* and *Good Research*. Figure 7 illustrates the impacted elements.

4.2 Case Study 2: Commuting System

The second case study is a GRL specification describing a commuting system. Figure 8 shows the impact (in purple) of changing the task *Take own car*, on both models *Commuting-Time* (Fig. 8(a)) and *Stakeholders* (Fig. 8(b)). The impacted elements are part of a strategy, called *Take own car, Alarm, Stairs only*.

5 Discussion

In what follows, we discuss the benefits and limitations of the proposed approach, then we compare it with related work.

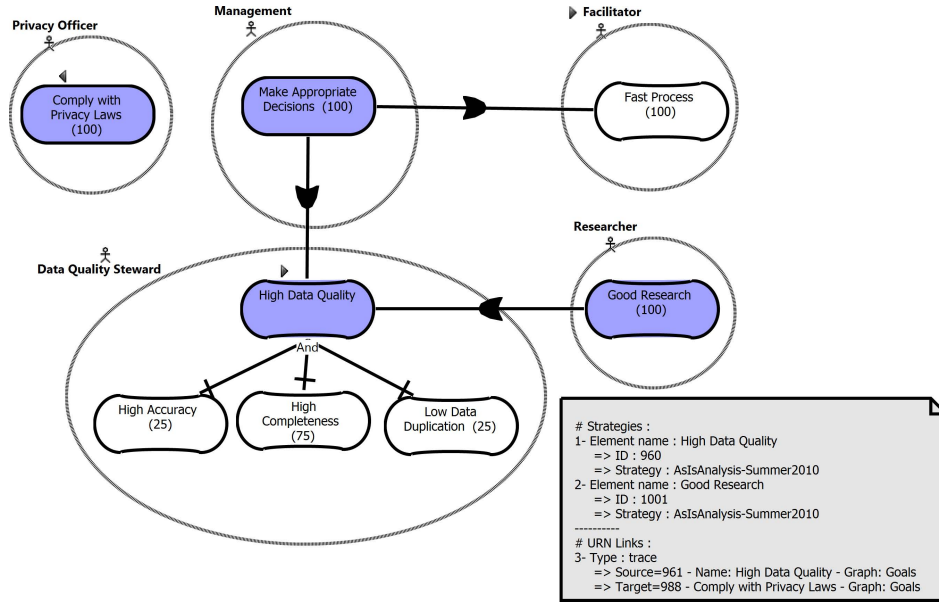


Fig. 7. Impacted elements of the second AEMS CIA task

5.1 General Benefits of the GRL-based CIA Approach

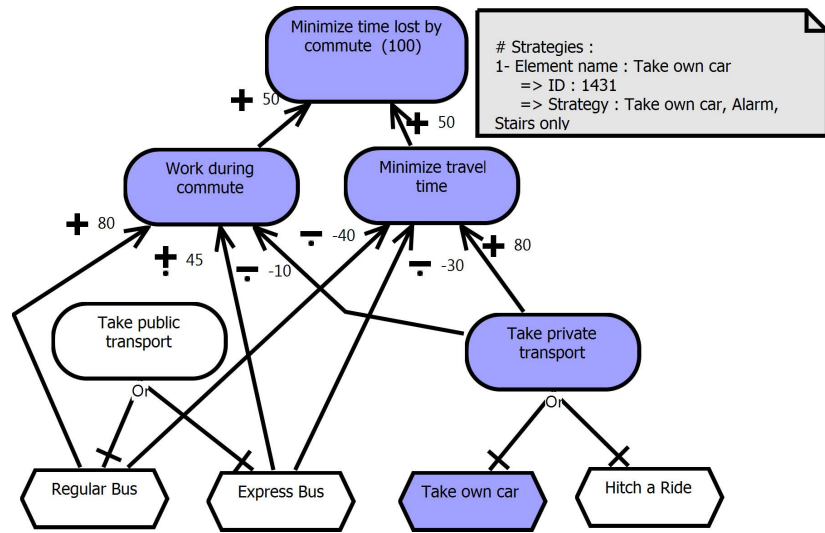
The presented GRL-based change impact analysis approach presents the following advantages:

- It helps maintainers and analysts answer “*what if... ?*” questions, and assess the consequences of changes in GRL specifications. Indeed, our approach provides an insight into how changes propagate within a GRL model, and across models (i.e., from GRL to GRL) through URN links. In addition, it allows for the identification of the impacted GRL strategies, if any. This would allow for reasoning about different alternatives, when it comes to implement changes in GRL models.
- Our approach is fully automated and covers the full GRL language constructs.
- We have chosen GRL as target language, given its status as an international standard, but our proposed approach can likely be adapted and applied to other goal-oriented languages such as i^* [1] and TROPOS [2].

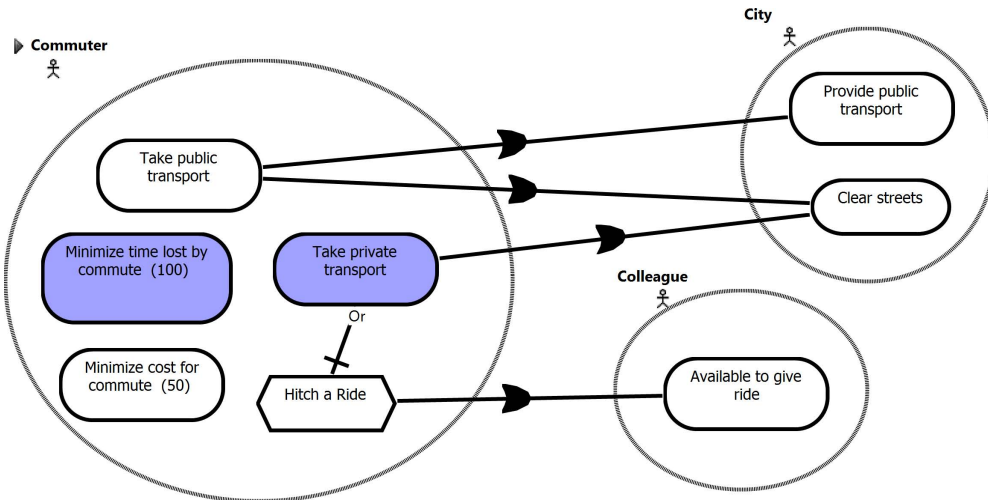
5.2 Limitations

The proposed CIA approach is subject to the following limitations:

- Our approach supports the evaluation of the impact of a single change at a time. Assessing the impact of simultaneous changes is left for future work.



(a) Impacted elements in the Commuting-Time Model



(b) Impacted elements in the Stakeholders Model

Fig. 8. Identification of impacted elements in two GRL models of the commuting case study

- We perform a single iteration to follow the involved URL links. The potentially impacted GRL elements are not used as a source/target to explore more URN connections, if any. However, we believe that implementing a transitive chain should take into account the semantics of the URN links (i.e., there should be a strong dependency that justifies the capture of the full ripple effect). This is out of the scope of this research.
- The applicability of our approach was demonstrated using two case studies and a mock system (not presented in this paper) only. Bigger case studies should provide a better assessment of the effectiveness of our proposed approach.

5.3 Comparison with related work

Change impact analysis [5] techniques have focused mainly on source code level [6] in order to help developers understand and maintain their programs. Less work has been devoted to change impact analysis in other software artifacts such as requirements and design models [11]. In what follows, we survey and compare existing goal-oriented CIA techniques with our proposed approach.

In a closely related work, Hassine [7] proposed a preliminary (and manual) CIA approach based on slicing GRL Model Dependency Graphs (GMDG). In this paper, we extend the approach by considering inter model propagation, GRL evaluation strategies, and URN links. We have also fully automated it. Cleland-Huang et al. [12] introduced a probabilistic approach for managing the impact of a change using a Softgoal Interdependency Graph (SIG) that describes non-functional requirements and their dependencies. This technique allows for the analysis of the impact of changes by retrieving links between classes affected by changes in the SIG graph. Our approach is based on the GRL graph structure and does not distinguish between functional and non-functional requirements.

Tanabe et al. [13] introduced a change management technique in AGORA. The technique aims at detecting conflicts when a new goal is added and checks the satisfaction of the parent goal, when a goal is deleted. Semantic information, described as goal characteristics such as security or usability, should be attached to goals to allow for the detection of conflicts. Our approach considers structural change (both addition and deletion) propagation within the same model and across many models, regardless the semantic aspect of the impacted goals. Lee et al. [14] proposed a goal-driven traceability technique for analyzing requirements, which connects goals and use cases through three different traceability relations (evolution, dependency, and satisfaction), which are stored as a matrix. Impacted entities can then be identified by applying a reachability analysis on the matrix. Our GRL-based approach builds a GRL model dependency graph (GRL) to represent explicit and implicit, e.g., contribution, dependencies between model elements. In addition, our approach identifies the potential changes in other model elements that are linked through user-defined URN links.

Ernst et al. [15] proposed an approach to find suitable solutions (that minimize the effort required to implement new solutions) as requirements change. Their approach [15] explores a Requirements Engineering Knowledge Base

(REKB), describing goals, tasks, refinements, and conflicts, in order to find new operations that are additionally required as a result of an unanticipated modification such as the addition of a new feature or the introduction of a new law. Our approach does simply spot potential impacted elements based on the GRL model structure and does not propose a solution to implement the change. In order to help developers identify where changes are required, Nakagawa et al. [16] proposed an approach based on the extraction of control loops, described as independent components that prevent the impact of a change from spreading outside them.

More recently, Grubb and Chechik [17] proposed an i*-based method to model the evolution of goal evaluations over time. Their proposed method integrates variability in intentions satisfaction (using qualitative values) over time allowing the stakeholders to understand and consider alternatives over time. In a closely related work to [17], Aprajita and Mussbacher [18] introduced Timed-GRL, an extension of the GRL standard, allowing for the capture and analysis of a set of changes to a goal model over time (using quantitative values such as concrete dates). Both the goal model and the expected changes are represented in one model. However, both approaches described in [17] and [18] focus only on the evolution of satisfactions values (qualitative and quantitative) and they do not consider the evolution of the goal model structure over time.

6 Conclusions and Future Work

In this paper, we have presented an automated GRL-based approach to change impact analysis. The proposed CIA approach allows maintainers and analysts understand how a change is propagated within a GRL model and across related GRL models (i.e., from GRL to GRL), linked using URN links. In addition, the approach allows for the identification of the potentially impacted GRL evaluation strategies. The approach has been implemented as a feature within the jUCMNav [8] tool.

As a future work, we plan to extend our approach to cover simultaneous GRL changes, and to assess the impact of such changes on related Use Case Maps (UCM) functional models.

Acknowledgment

The authors would like to acknowledge the support provided by the Deanship of Scientific Research at King Fahd University of Petroleum & Minerals for funding this work through project No. FT151004.

References

1. Yu, E.S.: Towards modelling and reasoning support for early-phase requirements engineering. In: Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on, IEEE (1997) 226–235

2. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the tropos methodology. *Eng. Appl. Artif. Intell.* **18** (March 2005) 159–171
3. ITU-T: Recommendation Z.151 (10/12), User Requirements Notation (URN) language definition, Geneva, Switzerland (2012)
4. van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. *IEEE Trans. Softw. Eng.* **26**(10) (October 2000) 978–1005
5. Bohner, S., Arnold, R.: Ieee computer society press. Los Alamitos, CA, USA (1996)
6. Li, B., Sun, X., Leung, H., Zhang, S.: A survey of code-based change impact analysis techniques. *Software Testing, Verification and Reliability* **23**(8) (2013) 613–646
7. Hassine, J.: Change impact analysis approach to grl models. In: *SOFTENG 2015: The First International Conference on Advances and Trends in Software Engineering, IARIA* (2015) 1–6
8. jUCMNav v7.0.0: jUCMNav Project (tool, documentation, and meta-model). <http://softwareengineering.ca/~jucmnav> (2014) Last accessed, June 2017.
9. Hassine, J., Alshayeb, M.: Measurement of actor external dependencies in GRL models. In Dalpiaz, F., Horkoff, J., eds.: *Proceedings of the Seventh International i* Workshop co-located with the 26th International Conference on Advanced Information Systems Engineering (CAiSE 2014)*, Thessaloniki, Greece, June 16-17, 2014. Volume 1157 of *CEUR Workshop Proceedings.*, CEUR-WS.org (2014)
10. Weiser, M.: Program slicing. In: *Proceedings of the 5th International Conference on Software Engineering. ICSE '81*, Piscataway, NJ, USA, IEEE Press (1981) 439–449
11. Lehnert, S.: A taxonomy for software change impact analysis. In: *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution*, ACM (2011) 41–50
12. Cleland-Huang, J., Settini, R., BenKhadra, O., Berezhanskaya, E., Christina, S.: Goal-centric traceability for managing non-functional requirements. In: *Proceedings of the 27th international conference on Software engineering*, ACM (2005) 362–371
13. Tanabe, D., Uno, K., Akemine, K., Yoshikawa, T., Kaiya, H., Saeki, M.: Supporting requirements change management in goal oriented analysis. In: *16th IEEE International Requirements Engineering (RE'08)*, IEEE (2008) 3–12
14. Lee, W.T., Deng, W.Y., Lee, J., Lee, S.J.: Change impact analysis with a goal-driven traceability-based approach. *International Journal of Intelligent Systems* **25**(8) (2010) 878–908
15. Ernst, N.A., Borgida, A., Jureta, I.: Finding incremental solutions for evolving requirements. In: *Requirements Engineering Conference (RE)*, 2011 19th IEEE International, IEEE (2011) 15–24
16. Nakagawa, H., Ohsuga, A., Honiden, S.: A goal model elaboration for localizing changes in software evolution. In: *21st IEEE International Requirements Engineering Conference (RE'2013)*, IEEE (2013) 155–164
17. Grubb, A.M., Chechik, M.: Looking into the crystal ball: Requirements evolution over time. In: *2016 IEEE 24th International Requirements Engineering Conference (RE)*. (Sept 2016) 86–95
18. Aprajita, Mussbacher, G.: Timedgrl: Specifying goal models over time. In: *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. (Sept 2016) 125–134