International Journal of Computers and Informatics (IJCI)

Vol. (1), No. (1)

IJCI

December 2022

المجلة الدولية للحاسبات والمعلوماتية

الإصدار (1)، العدد (1)

Requirements Elicitation for Software: Modeling Techniques

Jehad Mohamed Mousa
Master of Software Engineering, Software and Systems Project Manager, Ministry of Electricity, Iraq
Jehad.mousa@moelc.gov.iq

Amir Kamel Badr
Software Engineer, Ministry of Electricity, Iraq
ameerm@moelc.gov.iq

Abstract

This research aims to introduce Goal-Oriented Requirements Engineering (GORE), defining what is meant by a goal, the main differences between goal and requirement, also the types of goals and the sources of extracting these goals, in addition, the birth of goal modelling techniques and the reason behind using goal modelling, at last, the goal-oriented approaches, early and late requirements goal modelling techniques, this research tries to get out with the result of how goal modelling is very important in requirements engineering, in order to extract the goals and requirements in correspondence to business context, which in turn will aid in better analyses and extract the functions and processes in any organization or business.

Keywords: Goal-Oriented Requirements Engineering; Goal Modeling Techniques; Requirements Engineering.

1- Introduction

Recently the term goal has been used in requirements engineering techniques, and goals promoted into requirements engineering for a lot of causes, this is due to different and enormous activities and objectives of RE [14]. Goals in some RE are playing a fundamental role, researchers in this domain made a lot of efforts to prove the importance, significance and usefulness of this concept [2]. As RE pays attention to real-life observations, in order to translate these observations into mathematical specification language [1].

The importance of goal could be very powerful in; requirements acquisition, as to how to relate the theses goals to business situation as it gives a detailed description of this business and how it is operated, also it helps to clarify requirements, dealing with conflicts, aiding software design [13].

This research is organized as follows; the first section introduces what is a goal and the differences between goals and requirements, the second section a detailed description of types of goals and the source of extracting goals, the third part; elaboration on goal modelling techniques, which implies the birth of goal modelling techniques, the reason of using goal modelling, the benefits of using goal modelling, also when to use goal modelling, and goal-oriented approaches, in fourth part an illustration of early and late requirements goal modelling techniques.

2- Goals, Requirements, and Goals Activities

Any developed system can be judged or evaluated due to the degree of meeting the purpose it was done for, so in the first place identifying the purpose of what the system it was done for considered one of the main activities in software development. Goal-oriented approaches have risen to meet multi-stakeholder criteria and handle the consistency between their actions, in recent years the popularity of goal-oriented approaches has increased due to the inadequacy of traditional systems with more and more complex systems; most techniques focus on modelling and specifications of software alone; also non-functional requirements are left outside requirements specification mostly, in addition, the traditional techniques don't allow alternative system configurations. Goal-oriented approaches can handle the previously specified reasons.

To distinguish between goals and requirements, first identifying the goal, according to van Lamsweerde [15, 16], goal is an objective achieved by the system through the cooperation of agents in the software to be, which implies that the goal is the purpose of the system or software to be, another definition for goal by Anton, the goal is high-level objectives of the business, organization or system. Also, this definition implies that goal is related to business or any organization stakeholders, as it also implies that goals are purposes of system to be.

As the definitions of goal illustrated, this could lead to differentiate between goals and requirements, as a goal is a high-level objective for an organization or business, and an objective achieved by a system through the cooperation of agents in a software to be, as requirements is an objective achieved under the responsibility of one agent [15, 16], this could imply that goal is a broader term than requirements, and this is the reason of rise the techniques of goal modelling techniques.

Goal-oriented requirement engineering contains the following activities [14], first is goal elicitation, as stated in the literature identifying goals is not an easy task, as goals could be stated by different means like stakeholders or any other organizational sources of information and then these goals could be passed to software engineers, most frequently goals are implicit and therefore the elicitation process must take place. A preliminary analysis of the current system/organization is an important source of goal identification. This analysis can result in a list of problems and deficiencies that can be precisely formulated. As mentioned, goals could be elicited from many sources of information, these sources could be multiple kinds of documents. it is noted that stakeholders tend to express their requirements in terms of operations or actions, rather than goals. So, it makes sense to look for action words such as "schedule" or "reserve" when gathering requirements for a meeting scheduler system.

Also, another activity is goal refinement; once goals have been identified the aim is usually to refine them into progressively simpler goals until these goals can be easily operationalized and implemented. This process is usually done by asking the HOW questions and refining goals through AND/OR refinements. Many GORE approaches stress that when determining how a high-level goal can be refined, one needs to consider alternative ways of refining it to make sure that as many options as possible are explicitly represented in goal models and analyzed with respect to high-level criteria.

After goal refinement here come various types of goal analysis or sometimes called obstacle analysis, Potts identifies obstacles for specific goals by asking certain questions, Goal-based Requirements Analysis Method (GBRAM) uses the same method in managing obstacles, once an obstacle is identified, a set of events must be built, and these events identify why goal could be fail, so those built events could be an incarnation of obstacles for each goal [9]. it is noteworthy that a more specific goal leads to more preventing the obstacles to be. As obstacles can be formally identified as follows. Given a formal specification for a goal G, calculate the preconditions for obtaining the negation of G from the domain theory. Each obtained precondition is an obstacle [15].

Another activity is assigning goals to agents, this in KAOS approach, and agents are assigned leaf-level goals based on their abilities. The process is quite like GBRAM. KAOS permits requirements engineers to analyze substitution configurations of the boundary between the system-to-be and its environment through the use of OR responsibility links. That is why it is possible to compare several system configurations. In GBRAM several agents could be responsible for the same goal at different times.

3- Types of Goals

Goals can be categorized in many different ways according to literature, according to [13] goals can be classified as functional and non-functional goals, functional goals mean services that are expected to be the output of the system, as non-functional goals mean the qualities which the system will deliver, such as security, customizability, and flexibility as an example.

Goals could be classified as satisfaction goals and information goals, information goals can be defined as functional goals involved in keeping agents informed with object states, as satisfaction goals are concerned with replying to agents' questions. Goals also could be classified as Performance goals, which are specialized into time and space performance goals, the former being specialized in response time and throughput goals [15].

Security goals are specialized in confidentiality, integrity and availability goals [15]; the latter can be specialized in turn until reaching domain-specific security goals.

Another classification made is between soft goals, whose satisfaction cannot be established in a clear-cut sense [l92], and hard goals whose satisfaction can be established through verification techniques [15]. Soft goals are especially useful for comparing alternative goal refinements and choosing one that contributes the most to them.

Another classification axis is based on types of temporal behavior prescribed by the goal. [15]. Achieve (resp. cease) goals generate system behaviors, in that they require some target property to be eventually satisfied in some future state (resp. denied); maintain (resp. avoid) goals restrict behaviors, in that they require some target property to be permanently satisfied in every future state (resp. denied) unless some other property holds. Optimize goals and compare behaviors to favour those, which better ensure some soft target property.

Goal types and taxonomies are used to define heuristics for goal acquisition, goal refinement, requirements derivation, and semi-formal consistency/completeness checking [15], or to retrieve goal specifications in the context of specification reuse.

4- Goal Modeling Techniques

Goal-oriented approaches have been linked to the four RE activities as shown in the below table (1) linking to them goal-oriented approaches that can handle requirements within each activity.

Table (1) Goal-oriented approaches' link with RE activities [14]

RE Activity	*Goal Analysis Contribution*	*Goal-Oriented Approach*
• requirements elicitation	1. understanding the current organisational situation,	GOMS, Goal-based Workflow, *i**, EKD
	2. understanding the need for change	ISAC, F^3
• requirements negotiation	3. providing the deliberation context of the RE process	SIBYL, REMAP, The reasoning loop model
• requirements specification	4. relating business goals to functional and non-functional system components	KAOS, GBRAM , the NFR framework, the Goal-scenario coupling frame work
• requirements validation	5. validating system specifications against stakeholders' goals	GSN, GQM

One of the most popular examples for late requirements goal modeling techniques is the UML. UML was originally based on the object-oriented modeling technique whose aim is to provide a standard way to visualize the design of a system. UML has two types of views: systematic (or structural) view and dynamic (or behavioral) view. Despite being one of the most known techniques for modelling late requirements, UML suffers many disadvantages such as: it is so time-consuming, we can't identify exactly who benefits from UML, the UML diagram might be overcomplicated for the customer to understand and the emphasis most of the times is mainly on the design which annoys the developer in his work a lot.

To model early requirements properly, an agent-oriented approach has been defined to analyze such requirements and select the best alternatives that results later in achieving their goals. From the popular early requirements goal models are:

a) i* framework

I* framework has been proposed by Eric Yu [17] which later was discovered for many drawbacks that lead to the appearance of variants of i* which are Tropos and GRL [3]. The i* framework has two ways of representation: the graphical and the formal representations. As for the graphical representation, it has two main modeling components; the Strategic Dependency (SD) model and the Strategic Rationale (SR) model The SD model is concerned with describing mainly the dependency relationship between actors within the organization and as a result, it helps in understanding how a certain goal is embedded within the organization. Therefore, it always answers the "Whys" questions. On the other hand, the SR model is concerned with describing stakeholders' interests and how they might be addressed through various system configurations, and stakeholders' evaluation of various alternatives respecting their interests [4]. I* actors can be categorized into agents, positions, and roles. An agent occupies a position, a position covers a role. Consequently, an agent plays role covered by a certain position. Actors and their categorization can also be decomposed into another actor through the is-part-of relationship type. Actors have some intentional properties such as goal, belief, attributes, and commitment. I* framework provides several analyzing levels in terms of ability, workability, viability, and believability. I* models and analyzes security and privacy requirement using secure i* (SI*) [18].

The i* basic elements are: The intentional elements [3]: which are goals, soft goals, tasks, resources of an actor boundary. Links: which connect the i* model elements together through means-end, task-decomposition, or contribution links (positive (Make/Help links), negative (Hurt/ Break links) and the unknown. The satisfaction levels. Reasoning elements: they are represented through routines, rules, and beliefs.

For large and complex organizations, the i* graphical representation is not suitable as a result, i* formal notation is used. The formal notation uses the First Order Logic (FOL). Six predicates are used indicating the possible six analysis labels v= {FS (), PS (), FD (), PD (), C (), U ()} where:

FS(): represents the fully satisfied, PS(): stands for the partially satisfied, FD(): is the fully denied label, PD(): represents the partially denied, C(): stands for the conflict label and the U(): is the unknown label. The predicate holds when the label applies. Examples for propagation rules for labels.

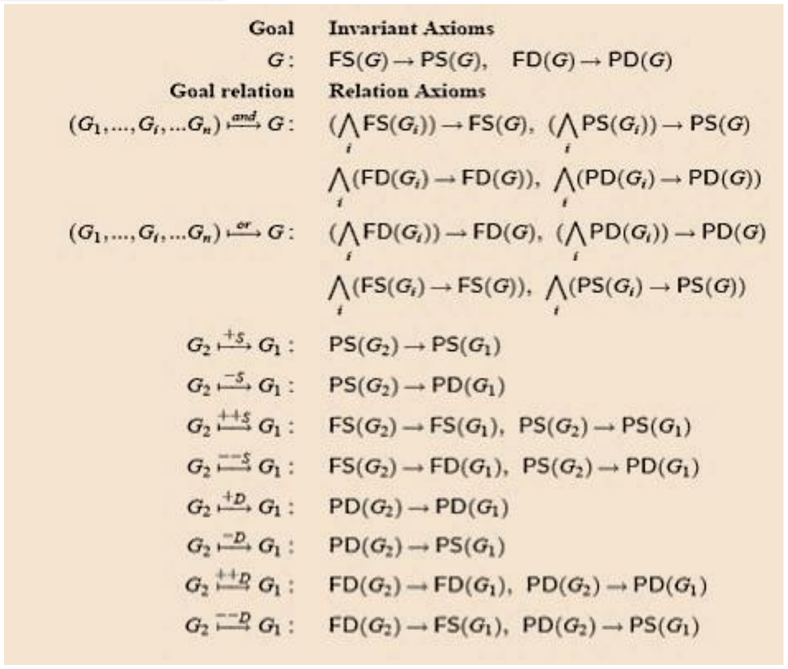
Despite the existence of many tools supporting i* language such as OME and REDEPEND, it always suffered from having some strange situations which mainly appears due to the incompleteness of the formalization of i*. The reason for this incompleteness is the existence of many confused situations, e.g., the “is-a” relation is used profusely, however, it is not defined as a i* constructor. Moreover, many incomplete definitions exist, e.g., no indication about the type and the number of roots in the internal decomposition of an actor. In addition to the presences of some ambiguous definitions, e.g., the dependency link must have two different importance degrees each implies one of the involved actors (depender, dependee).

b) Tropos

Tropos is an agent-oriented software development methodology. It adopts i* model. Tropos supports four phases of software development, which are early requirements, late requirements, architectural design, and detailed design. The early requirement phase is intended to understand the organizational context that the system-to-be will be built on. The Late requirement phase is concerned with defining the system-to-be's functional and non-functional requirements.

The architectural design is concerned with defining the system's global architecture, whereas the detailed design concerned with defining the behavior of each software component in more details. Tropos can represent organizational goals either graphically or formally [8]. Tropos models and analyzes security and privacy requirement using secure Tropos [19].

Tropos graphical representation has two diagrams for modeling and analysis of organizational requirements and goals, which are the actor diagram and the rationale diagram. The actor diagram, similar to SD model in i* while the rationale diagram is similar to SR model in i*. Tropos basic elements are the same as those of i* differing only in the links types, where Tropos provides AND/OR decomposition link instead of the task- decomposition of i*. Tropos Formal Representation is preferred to model large and complex organizations requirements. It uses the FOL of that of i*. Tropos forward propagation rules are found in figure 1. Backward propagation rules are found in figure 2[12].



Goal	Invariant Axioms
$G:$	$\mathrm{FS}(G) \rightarrow \mathrm{PS}(G), \quad \mathrm{FD}(G) \rightarrow \mathrm{PD}(G)$
Goal relation	**Relation Axioms**
$(G_1, \ldots, G_i, \ldots G_n) \xmapsto{and} G:$	$(\bigwedge_i \mathrm{FS}(G_i)) \rightarrow \mathrm{FS}(G), \ (\bigwedge_i \mathrm{PS}(G_i)) \rightarrow \mathrm{PS}(G)$
	$\bigwedge_i (\mathrm{FD}(G_i) \rightarrow \mathrm{FD}(G)), \ \bigwedge_i (\mathrm{PD}(G_i) \rightarrow \mathrm{PD}(G))$
$(G_1, \ldots, G_i, \ldots G_n) \xmapsto{or} G:$	$(\bigwedge_i \mathrm{FD}(G_i)) \rightarrow \mathrm{FD}(G), \ (\bigwedge_i \mathrm{PD}(G_i)) \rightarrow \mathrm{PD}(G)$
	$\bigwedge_i (\mathrm{FS}(G_i) \rightarrow \mathrm{FS}(G)), \ \bigwedge_i (\mathrm{PS}(G_i) \rightarrow \mathrm{PS}(G))$
$G_2 \xmapsto{+S} G_1:$	$\mathrm{PS}(G_2) \rightarrow \mathrm{PS}(G_1)$
$G_2 \xmapsto{-S} G_1:$	$\mathrm{PS}(G_2) \rightarrow \mathrm{PD}(G_1)$
$G_2 \xmapsto{++S} G_1:$	$\mathrm{FS}(G_2) \rightarrow \mathrm{FS}(G_1), \ \mathrm{PS}(G_2) \rightarrow \mathrm{PS}(G_1)$
$G_2 \xmapsto{--S} G_1:$	$\mathrm{FS}(G_2) \rightarrow \mathrm{FD}(G_1), \ \mathrm{PS}(G_2) \rightarrow \mathrm{PD}(G_1)$
$G_2 \xmapsto{+D} G_1:$	$\mathrm{PD}(G_2) \rightarrow \mathrm{PD}(G_1)$
$G_2 \xmapsto{-D} G_1:$	$\mathrm{PD}(G_2) \rightarrow \mathrm{PS}(G_1)$
$G_2 \xmapsto{++D} G_1:$	$\mathrm{FD}(G_2) \rightarrow \mathrm{FD}(G_1), \ \mathrm{PD}(G_2) \rightarrow \mathrm{PD}(G_1)$
$G_2 \xmapsto{--D} G_1:$	$\mathrm{FD}(G_2) \rightarrow \mathrm{FS}(G_1), \ \mathrm{PD}(G_2) \rightarrow \mathrm{PS}(G_1)$

Fig. (1) Tropos forward propagation rules

Tropos forward and backward propagation is supported by the Goal Reasoning Tool (Gr-Tool1). The Gr-Tool is a graphical tool for representing goal models and applying the required analysis algorithm. GOALSOLVE and GOALMINSOLVE tools are implemented to support the backward propagation first and second approaches.

c) Goal-oriented Requirement Language (GRL)

GRL is an agent-oriented and goal-oriented modelling language that supports reasoning about non-functional requirements and quality attributes. It is influenced by both the i* and the NFR frameworks for specifying non-functional requirements. Recently, the International Telecommunication Union (ITU-T) as a part of the User Requirements Notation [5] standardizes GRL.

$$FS(G) \rightarrow \begin{cases} \bigwedge_i FS(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{and} G \\ \bigvee_i FS(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{or} G \\ FS(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{++S} G \\ FD(G_i) & \text{For every } R_i : G_i \xrightarrow{--D} G \end{cases} \qquad FD(G) \rightarrow \begin{cases} \bigvee_i FD(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{and} G \\ \bigwedge_i FD(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{or} G \\ FD(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{++D} G \\ FS(G_i) & \text{For every } R_i : G_i \xrightarrow{--S} G \end{cases}$$

$$PS(G) \rightarrow \begin{cases} \bigwedge_i PS(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{and} G \\ \bigvee_i PS(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{or} G \\ PS(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{++S} G \\ PD(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{--D} G \\ PS(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{+S} G \\ PD(G_i) & \text{For every } R_i : G_i \xrightarrow{-D} G \end{cases} \qquad PD(G) \rightarrow \begin{cases} \bigvee_i PD(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{and} G \\ \bigwedge_i PD(G_i) \vee & \text{If } (G_1,\ldots,G_i,\ldots G_n) \xrightarrow{or} G \\ PD(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{++D} G \\ PS(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{--S} G \\ PD(G_i) \quad \vee & \text{For every } R_i : G_i \xrightarrow{+D} G \\ PS(G_i) & \text{For every } R_i : G_i \xrightarrow{-S} G \end{cases}$$

Fig. 2. Tropos backward propagation rule

User Requirements Notation (URN) is used to support all the RE phases. It allows requirement engineers to elicit and specify requirements then analyze such requirements to ensure its completeness and correctness. URN consists of two complementary languages which are Goal-oriented Requirement Language (GRL) and Use Case Map (UCM) [6].

The benefit of using GRL is that it can be integrated with scenario notation and can define a clear separation between its elements and their graphical representation. Moreover, it enables a scalable and consistent representation of multiple diagrams for the same goal model.

It has been influenced by i* language in: GRL goal model has three basic concepts which are actors, intentional elements, and links, GRL links types are decomposition, contribution, dependencies, The main differences respecting i* are that GRL offer a new link type namely, correlation link. Correlation link describes the side effects of an element on another element.

Also, GRL offers constructors for enabling relationships with external elements. Moreover, the use of URN links and metadata for enabling linking GRL with UCM elements. However, GRL supports only one type of actor, whereas i* support the notations of role, agent, and position.

There exist many GRL tools; some of them are jUCMNav and OME. JUCMNav tool is an Eclipse plug-in for the creation, analysis, and transformation of URN models.

Amyot presented a tool providing a lightweight profile for GRL that enables creating a goal model in i* style. Such profile supplements GRL with i* missing concepts using the advantage of URN links and metadata. Moreover, it restricts the usage of GRL to i* through the usage of UML's Object Constraint Language (OCL). The tool is implemented in the jUCMNav tool.

The goal satisfaction analysis procedure is applied on a goal model to select alternatives that aim to satisfy the desired goal. Goal satisfaction is either forward or backward propagation whether it is qualitative or quantitative [7].

The forward propagation starts by initializing a set of alternatives with a satisfaction value, and then propagates such values upward iteratively through links and forward propagation rules until reaching the top goals [20].

The backward propagation starts by initializing the top desired goals with satisfaction values, and then propagates such values downward iteratively through links and backward propagation rules. i* framework analysis supports both forward and backward propagation analysis on both qualitative and quantitative.

Tropos analysis supports both forward and backward propagation analysis on both qualitative and quantitative. It solves the back propagation problem using two approaches. In the first approach, it reduces the problem of the backward propagation to that of the propositional satisfiability (SAT).

In the second approach, Tropos tries to find the set of alternatives with minimum cost that achieve the desired top goals using the Minimum-Weight Propositional Satisfiability (MW-SAT). GRL analysis supports both forward and backward propagation analysis on qualitative, quantitative and hybrid analysis.

In real world applications, most of the goal models suffer from uncertainty and this is due to having gaps in the knowledge domain, disagreement between stakeholders or the presence of uncertainty over requirement details. Therefore, it is important to handle uncertainty since ignoring uncertainty may lead to selecting alternatives that may not be sufficient to achieve the desired goals or eliminating viable alternatives.

Horkoff proposed a semi-automated tool to handle uncertainty in early requirements represented in i* using MAVO framework in addition to her proposed formal analysis formula based on the forward qualitative analysis [10].

Initially, Horkoff goal model analysis methodology constructs a set of all possible concretizations where each of them represents a concrete goal model results from resolving a specific uncertainty requirement. Then, for each concretization she applies forward qualitative analysis. Afterwards the user is allowed to select her choices for each top goal, which is then checked for simultaneous availability in any of the possible set of concretizations. Reduced uncertainty respecting the user acceptability and the domain consistency is evaluated for the concretization results in the simultaneous achievable choices. Any appearing changes will lead to further analysis.

The results for contextual goal modeling and reasoning framework defined that more than one alternative can satisfy the top desired goal. The decision of selecting among them depends on the applied context. Thus, it is important to enrich goal models with context. Consequently, Raian proposed a goal-oriented requirement engineering modeling and reasoning framework for systems operating under various contexts [11].

The framework relates context and goals using Tropos. Then, it analyzes all contexts to identify ways for verifying them. To derive requirements reflecting a certain context automatically, the framework proposes two reasoning techniques, namely, design time and runtime reasoning techniques. The runtime reasoning technique concerns deriving goal model variants reflecting context and user priorities. The design time reasoning technique concerns deriving requirements for the system-to-be with minimum cost and valid in all considered contexts.

5- Conclusion

It has been realized that goal-modeling techniques are being preferred than traditional methods such as object-oriented approaches. This research showed that requirements are considered to be a subset of goals and in return goals have been the major umbrella that concerns everyone for modeling. Moreover, this research found that late requirements modeling hasn’t got so much interest such as that of early requirements modeling. An illustration of three well known early requirements models which are i* framework, Tropos and GRL. Also, the research addressed in brief UML, which is considered the most popular late requirements modeling technique.

References

[1] P. Save, “Classification of Research Efforts in Requirements Engineering", ACM Computing Surveys, Vol. 29, No. 4, 1997, pp. 315-321.

[2] Yu ESK, Mylopoulos J, “Why goal-oriented requirements engineering”, In Dubois E, Opdahl AL, Pohl K (eds) Proceedings of the 4th international workshop on requirements engineering: foundation for software quality (RESFQ 1998). Presses Universitaires de Namur, Namur.

[3] C. P. Ayala, C. Cares, J. P. Carvallo, G. Grau, M. Haya, G. Salazar, X. Franch, E. Mayol, C. Quer, “A Comparative Analysis of i*-Based Agent-Oriented Modeling Languages”, 7th International Conference on Software Engineering and Knowledge Engineering, pp. 43-50, 2005.

[4] D. Quartel, W. Engelsman, H. Jonkers, M. van Sinderen, “A goal-oriented requirements modeling language for enterprise architecture”, Enterprise Distributed Object Computing Conference, pp. 3-13, EDOC '09, IEEE International, 2009.

[5] D. Amyot, J. Horkoff, D. Gross, G. Mussbacher, “A Lightweight GRL Profile for i* Modeling”, ER 2009 Workshops on Advances in Conceptual Modeling, Lecture Notes, pp. 254-264, 2009.

[6] D. Amyot and G. Mussbacher, "Development of Telecommunications Standards and Services with the User Requirements Notation".

[7] S. Ghanavati, D. Amyot, L. Peyton, “Compliance Analysis Based on a Goal-oriented Requirement Language Evaluation Methodology”, Requirements Engineering Conference, RE '09. 17th IEEE International, 2009.

[8] P. Giorgini, J. Mylopoulos, R. Sebastiani, “Goal-Oriented Requirements Analysis and Reasoning in the Tropos Methodology”, International Journal Engineering Applications of Artificial Intelligence, Volume 18, issue 2, pp. 159-171, 2005.

[9] J. Horkoff, E. Yu, “Analyzing Goal Models – Different Approaches and How to Choose Among Them”, ACM Symposium on Applied Computing, pp. 675-682, 2011.

[10] J. Horkoff, R. Salay, M. Chechik, D Sandro, "Supporting Early Decision-Making in the Presence of Uncertainty”, Requirements Engineering Conference (RE), 2014 IEEE 22nd International, pp. 33-42, 2014.

[11] A. Raian, F. Dalpiaz, P. Giorgini, “A Goal-based Framework for Contextual Requirements Modeling and Analysis”, Requirements Engineering Journal, Volume 15, Issue 4, November 2010, pp. 439-458, 2010.

[12] J. Horkoff and E. Yu, “Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach”

[13] Torkar, R., Gorschek, T., Feldt, R., Raja, U. A., & Kamran, K., 2009. Requirements traceability state-of-the-art: A systematic review and industry case study. IST Journal.

[14] Kavakli, E. a., 2003, "Goal Driven Requirements Engineering Evaluation of Current Methods", Proceedings of the 8th CAiSE/IFIP8., (p. 16).

[15] VAn Lamsweerde, A., 2001, "Goal-Oriented Requirements Engineering: A Guided Tour. Requirements Engineering", Proceedings of 5th IEEE international Symposium on IEEE (pp. 249-262). IEEE.

[16] Van Lamsweerde, A., 2000, "Requirements engineering in the year 00: A research perspective", Proceedings of the 22nd international conference on Software engineering. (pp. 5-19). ACM.

[17] Yu, E. S., 1997, "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering", In Requirements Engineering Proceedings of the Third IEEE International Symposium (pp. 226-235). IEEE.

[18] Kiyavitskaya, N., & Zannone, N., 2008, "Requirements Model Generation to Support Requirements Elicitation: The Secure Tropos Experience", Automated Software Engineering (pp. 149-173). Springer.

[19] Massaccia, F., & Zannone, M. P. 2005, "Using a Security Requirements Engineering Methodology in Practice: The Compliance with the Italian Data Protection Legislation", Computer Standards & Interfaces (pp. 445-455), Elsevier.

[20] Yu, J. H., 2010, "Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach", Conceptual Modeling-ER (pp. 59-75), Springer Berlin Heidelberg.