Doctoral Dissertation (Shinshu University)

On the development of a web-based toolkit for supporting reviews of the quality and contents of iStar requirements models

September 2017

MEJRI HAJER

Abstract

Choosing a paradigm or methodology to teach when incorporating the requirements engineering (RE) subject in engineering curriculum is delicate and challenging as a fundamental question may arise: Which of the existent methodologies is the best and the most suitable for this "holy" mission *i.e.* educational situation?

In this context, the *i** framework and its modeling language (the *i** language), a widespread and popular goal- and agent- oriented approach, has witnessed efforts on its introduction in engineering levels worldwide.

Despite its simplicity, as the *i** language is constituted basically by a simple set of graphical constructs which can be used in exactly two types of diagrams: the Strategic Dependency (SD) and Strategic Rationale (SR), as well as the existence of the *i** wiki portal that provides a collection of modeling guidelines, rules and best practices; there is still misconception, misinterpretation and misrepresentation depicted in novice learners' built diagrams which suggest their lack of the *i** modeling language syntax and semantics grasping.

Like any other existent modeling technique, it is consolidated by free computer-assisted tools that offer a number of functionalities to help users, in our case students particularly the novice learners, sketch their requirements models as well as check their quality. However, the freely accessible tools that we investigated so far do not cover neither implement the complete list of *i** rule checks and this may hinder the production of good quality models. To put it more simply, we gathered evidences of these tools' limitations which indicate that there is no complete and solid support for the *i** framework's novice learners on the model quality checking side.

Accordingly, we reviewed a decent portion of the existing literature and we discovered that only little work was devoted to enhance the syntactical quality of the *i** models and unfortunately, to our knowledge, the model content validation was left totally neglected. Obviously, more effort on model checking and validation is needed to allow better support and experience of the *i** goal modeling's novice learners.

The contribution of this thesis is twofold. It addresses the aforementioned issues as it adds the necessary support facilities by developing a web-based toolkit which is model quality checking- and content validation- oriented tools. Thus, it aims at guiding new users to review and revise the quality and contents of their diagrams without the help of a human assistant nearby.

The development process was performed individually in series of steps, addressing first the construction of model quality checker called *i*Check* in which we tried to cover the wiki-based and derived list of

checks for both types of *i** models by returning clear feedback consolidated with suggestions consisting in correction-oriented GIF animations and in a like manner, the building of *i** model content validation system aka GENERAT*i**ON which returns to its users the model's structural and informational (textual annotations) content in a table of contents (TOC) format.

The effectiveness of our toolkit was investigated through several experimental tests in which we asked new learners to debug a series of *i** diagrams to evaluate how each tool can guide them to locate and correct defects in given models. Results were positive and promising indicating the benefits that novices can get when having such tools around since it is unlikely that an instructor or assistant will always be nearby to offer advice on diagram construction and content validation.

Acknowledgments

It is a great pleasure to thank the many people who supported me along my Ph.D. studies route.

Words will never be enough to express my gratitude to my advisor Pauline Naomi Kawamoto for her constant and precious guidance throughout my doctoral program, for the time spent on discussing and accurately reviewing all the research work of this thesis. Besides her scientific competence, I enjoyed a lot her expertise, her patience, her understanding, her comprehension in my darkest hours and her positive attitude to make a friendly work atmosphere. I learned much more than scientific concepts from her, and her enthusiasm has been fundamental to overcome all the difficulties which arose in these three years.
Thank you so much for having your office always Open for me! Also, you helped me in many things outside my studies which are related to my everyday life in Nagano. Thank you very much!

I would like to thank the internal and external reviewers of this thesis, for their careful reading and for their valuable comments, suggestions and detailed feedback which served as a very useful inspiration for refining the final version of the thesis.

I really appreciate the opportunity to collaborate with Kawamoto laboratory past and present members. I found very valuable colleagues with whom I shared work and entertainment. I would like to thank all the students participating in the experimental studies conducted in the context of my research without whom, the evaluation of the research results would not be possible.

Unreserved thanks to all who made bearable the challenging moments through their friendship, care and support.

Finally, I am eternally grateful to my parents and siblings for their unflagging love and support throughout my life. Many Thanks to all of you, this thesis would have no sense without you all!

On the development of a web-based toolkit for supporting reviews of the quality and contents of iStar requirements models

Contents

List of Figures

List of tables

Chapter 1. Introduction

This chapter provides an introduction to this PhD research work. Section 1.1 describes the context of the present research, section 1.2 focuses on stating the problems addressed in this thesis, section 1.3 highlights the research goals, section 1.4 briefly introduces and discusses the proposed solutions for each mentioned issue *i.e.* our research contributions and finally, section 1.5 outlines this document structure.

1.1 Research context

Throughout the years, a significant number of approaches and methodologies have been formulated and proposed in order to assist the complex and the successful information systems construction. Particularly, there exist some efforts which concentrate on obtaining a richer understanding of the organizational environment before even starting the development of the future software systems [1]. In this sense, a considerable amount of attention has been paid to the early stage of Requirements Engineering (RE) and which resulted in developing numerous techniques and modeling languages.

During this phase, namely the early RE, the requirements engineer attempts to determine: a) the intentions (goals) and the social relationships (dependencies) of the different actors, b) the role of the software system-to-be in the organizational context by identifying the reason 'the *why*' *i.e.* the need behind its construction, c) the impact of the system in the performance of the organizational processes, and d) the alternatives that may be considered to meet the original objectives. Hence, the captured knowledge will permit building a software system that works harmoniously within the organization processes.

In this context, the *i** (iStar) framework [2], a well-founded and widespread RE framework and modeling language in use today since it allows expressing the intentional and the explicit social relations between actors as well as describing the internal behaviors which are either behind or working to satisfy the actors dependencies.

Its modeling language blends concepts coming from both goal-oriented RE (*e.g.*, goal) and agent-oriented RE (*e.g.*, agent). As a goal-oriented language, it aims to include the '*why*' of the decisions taken during system development. As an agent-oriented language, it includes the notion of agent and even more generally, the notion of 'actor'.

This framework is gaining a lot of attention, particularly, in being one of the top candidates for introducing requirements engineering (discipline) methodologies in undergraduate and graduate engineering programs worldwide [3, 22, 23, 24, 25, 26, 27, 28, 29]. Despite its simple set of graphical constructs, mastering the techniques of expressing system requirements using the *i** language is not straightforward for novice learners aka students. It typically requires some practical training time to learn and grasp its concepts [3] and subsequently model the world under study in a correct way which proves that novices understand the domain knowledge and information that are shown in their visual depiction.

1.2 Problems Statements

In this section, we present the main problems that we addressed in this work, namely: a) the confusion and misunderstanding that novice learners may face in their *i** modeling and learning journey and which mainly stem from the incomplete and limited checking features provided by the freely available tools and b) the absence of *i** models' contents review and validation approaches.

1.2.1 Checking limitations of the existent *i** modeling tools

The *i** framework has been increasingly used by the requirements engineering (RE) community generally and in particular, it is has been utilized to introduce and teach the RE discipline along other related courses such as System Information Analysis and Process Modeling in engineering programs. This fact is encouraged by the existence of a spectrum of *i** language variants' family, state-of-the-art computer-aided modeling systems and most importantly a dedicated *i** Wiki page [4], which is a rich collection of modelling constructs, documentation, conventions and guidelines and best practices. However, applications of some of its concepts are not straightforward for the novice learner and will require practical training time and intensive effort to master the techniques [3].

Now, in surveying the freely available *i** tools [5][6][7][8] that could be used in a classroom setting for this type of training, we found that none of them provides nor implements the full set of (in-depth) syntax checking features for detecting errors in the application of the core specification and modeling concepts. Even when a rule violation is detected in a model, the error message generated sometimes does not provide adequate and accurate information to direct and guide the beginner on how to fix the problem at hand. In other words, the studied tools are only equipped with limited checking features sometimes with poor, ambiguous or even misleading error message explanation which is amenable to originate different

interpretations and also can create confusion and misunderstanding for students studying alone or even in a large classroom setting where it is unlikely that an instructor or teacher assistant will always be nearby to offer advice on diagram construction and to review its content for subsequent validation purpose and if possible, it may be error-prone, time consuming and tedious task for him/her. To sum up, this problem is important in the specific case of *i** especially when models become large and complex, checking rules compliance becomes fundamental and difficult to achieve without the support of the automatic tools particularly when the produced models would reveal quality deficits which means much more effort is expected to fetch errors, correct them as well as unlearn the acquired modeling bad practices and habits.

Figure 1.1 presents an *i** model which contains and highlights some examples of *i** rules violation and bad modeling practices.

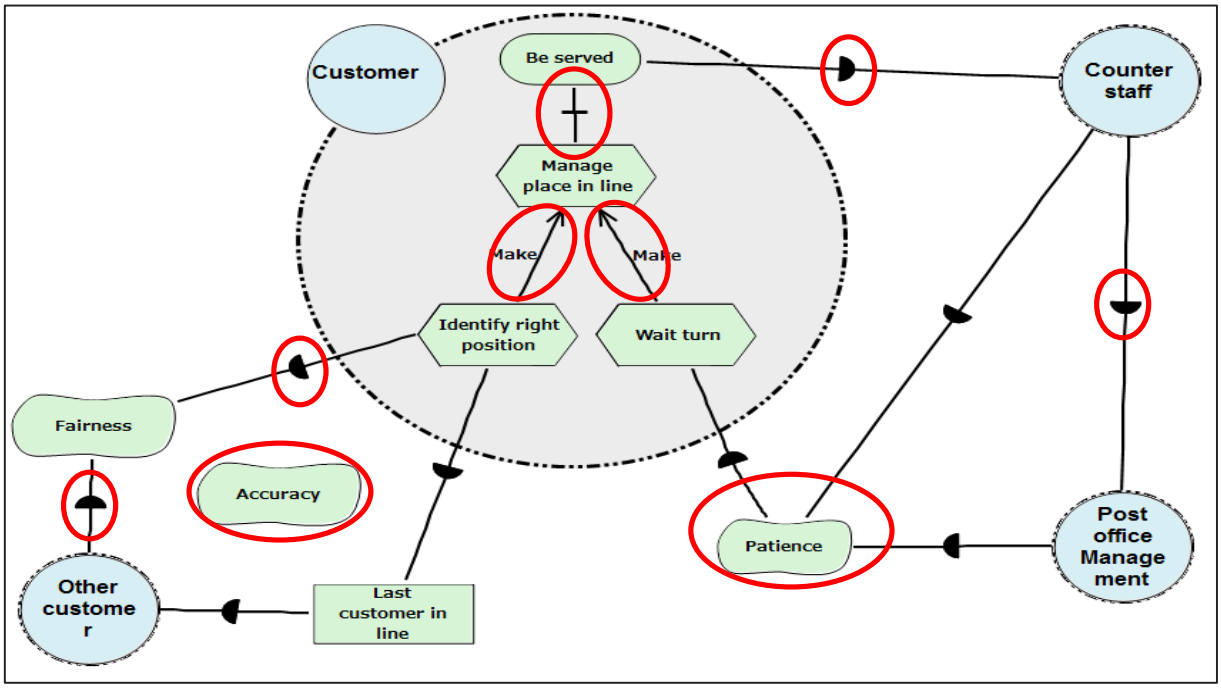


Figure 1.1: Excerpt from an *i model including some examples of defects proving the bad practice**

1.2.2 The lack of *i** model content validation

The main challenge in systems development is that of building the *right* system – one that meets the user needs at a reasonable cost. The key to achieving this goal lies in the early stages of systems development, *i.e.* during the requirements engineering process where the requirements of the system to be built are analyzed and often described using a conceptual model. It is essential that the constructed model of the system correctly represents the piece of reality under consideration *i.e.* domain knowledge.

The process of ensuring that a model possesses these qualities (coherence between the results and their original specifications) is called validation which is often an informal process where different stakeholders participate including people with limited knowledge of the modelling activity and the system design. In other words, the audience of these models ranges from well-trained requirements engineers and developers to casual staff members who are inexperienced in terms of modeling languages and may have severe difficulties in reviewing, understanding, evaluating and subsequently validating a model.

Accordingly, one could argue that the amount of information included in *i** both conceptual models, namely the SD (Strategic Dependency) and SR (strategic Rationale) may be a subject to validation and compliance's review and checking. Hence, in our interest, the novice learners of the *i** graphical notation who are inexperienced with the created requirements models may need the chance to validate the structure, behavior and the informational content of their sketched models.

1.3 Research goals

As the title of this thesis "On the development of a web-based toolkit for supporting reviews of the quality and contents of iStar requirements models" suggests, the aim of our research is to face quality issues on models raised by the usage of the *i** (iStar) as well as its provided tools. To do so, we fixed two specific objectives for this work:

1) To complement the existing tools' checking features by constructing a thorough online *i** model quality checker, called *i**Check, of almost all specification concepts and modeling rules and conventions presented in the guidelines of the *i** Wiki guide. The tool offers different kinds of feedback which aim to help the beginner recognize modeling mistakes and deficits at an early building stage as much as possible and offer him corrective scenarios for each detected flaw.
2) To allow the beginners, given their limited knowledge and skills (not enough experience dealing with the *i** modeling language) to review, evaluate and validate their built *i** goal models. To this purpose, we built a web-based application which we named as GENERAT*i**ON. Our tool permits the generation of a table of contents from an input XML-based representation of a certain *i** model for the purpose that consists in validating the "submitted" model content.

1.4 Proposed solutions

In the context of achieving the goals of this research work, this thesis work provides the following contributions consisting namely in developing a free web-based toolkit.

1.4.1 *i*Check*: an online free tool to detect and suggest corrections to *i** models defects

One contribution of this research work is the proposal and the development of an online tool which aims to detect *i** models defects and return helpful feedback to novice learners. *i*Check* covers a predefined list of checks to support the deep and extensive evaluation thus the checking of the created diagrams. The idea is to let students (beginners to the *i** framework) upload the istarML file, an XML-based version of their built *i** diagrams, and request the online checking service provided by our tool in order to subsequently obtain feedback for any existing modeling flaw. To sum up, we aim to help the novices consider and improve their models quality.

1.4.2 GENERAT*i**ON: an approach for model content review and validation

With the objective of addressing the problem of validating *i** goal models content and helping the students and novice users review and understand what it has been depicted in the diagramming format they built; an approach for generating table of contents (TOC), *i.e.* summary, from an input *i** model is proposed and accordingly, a web application was developed for this purpose. It consists in translating and verbalizing, corresponding to some semantic features of the *i** modeling language, the structural *i.e.* different types of relationships as well as the textually annotated *i** diagram elements aka the constructs' labels. As a result, the manual step of model content validation is reduced to just one single click.

1.5 Thesis structure

The organization of this thesis is presented in:

Chapter 1. Introduction

This introductory chapter highlights the research context and provides a brief overview of the issues analyzed in this thesis, research goals and the proposed solutions. Put differently, it presents the research work roadmap.

Chapter 2. Background baselines:

This chapter was intended to give the background of the dissertation, by paving the way for the Requirements Engineering overview, the existing goal modeling approaches to finally focus on the *i** framework which is the center of our research interest.

Chapter 3. Supporting the iStar Model Quality Review

In this chapter there is a description of the first system we developed in order to alleviate the first motivational issue *i.e.* limited checking features of the existent *i** modeling tools along with experimental evaluation results as well as some observations.

Chapter 4. Supporting the iStar Model Content Review

This chapter presents the solution we proposed in order to address the second problem of this research work namely the lack of *i** model contents validation against facts and domain knowledge.

Chapter 5. Conclusion and future work:

This chapter summarizes the contributions of the thesis and gives insights for future work.

Chapter 2. Background baselines

This chapter has the objective of providing the necessary background knowledge to understand the remainder of the thesis. It describes the relevant concepts, the theoretical foundations and the context of the present work. The chapter is organized as follows: Section 2.1 introduces an overview of the Requirement Engineering (RE) phase. Section 2.2 focuses on the early RE area. Then, section 2.3 presents the Goal modeling approach. Section 2.4 is concerned with providing an overview of the *i** framework.

2.1 Requirements Engineering

Requirements Engineering (RE) is a specific discipline of the software engineering. Being itself a process, it is recognized as being the most critical phases in software development.
Its main task is to generate correct specifications that clearly, unambiguously, consistently and compactly, describe the behavior of the system-to-be. Thus, it seeks to minimize problems related to systems development. Put differently, errors made during this stage may have negative effects on subsequent development steps and on the quality of the resulting software.

In the RE process, we can distinguish two groups of activities. The first group is related to the requirements development and the second group gathers activities classified as requirements management activities. The former group contains activities for requirements elicitation, analysis, specification, and validation. On the other hand, requirements management covers establishing and maintaining an agreement with stakeholders on the requirements, controlling the baseline requirements and their changes, and finally keeping requirements consistent with plans and work products.

This process deals not only with technical knowledge but also with organizational, managerial, economic and social issues. In this sense, a requirement specification should include not only software

specifications, but also any kind of information describing the context in which the intended system will function.

According to [9], Requirements Engineering has the following objectives (among others):

(a) Proposing communication techniques that facilitate the acquisition of information;
(b) Developing techniques and tools that result in appropriate and precise specifications of requirements;
(c) Considering alternatives in the specification of requirements and
(d) Developing executable specifications to help to speed up the production of a prototype.

Recently, scholars have made a separation of the requirements stage between early stage RE (organizational analysis) and late stage RE (requirements analysis) [10]. This separation of concerns produces a differentiation between research techniques that focus on social concerns (applicable at the early RE stage) and those that focus on technical concerns (for late RE).

With attention to early RE, the activities in this phase are typically informal and address organizational or non-functional requirements. The emphasis is on understanding the motivation and rationale that underlie system requirements.

In contrast, the late-phase RE usually focuses on completeness, consistency, and automated (formal) verification of requirements. It is concerned with the production of a requirement document such that the resulting system would be adequately specified and constrained in a contractual setting.

Since in this thesis we concentrate on the early Requirements Engineering, this latter will be the subject of the next section.

2.2 Early Requirements Engineering

The early requirements Engineering [10] is a new research area and it is considered as one of the most important and difficult phases of the software development process. In this phase, the requirements engineer attempts to understand the organizational context, the goals and social dependencies of its stakeholders in order to have the appropriate information to develop the future information system. This phase demands critical interactions with the users since a misunderstanding at this point may lead to expensive errors during later development stages. Not surprisingly, several approaches have been devoted to developing languages and analysis techniques for early requirements analysis (NFR, KAOS, *i**, Tropos).

The main feature of these techniques is the analysis and understanding of the organizational processes before starting the construction of an information system.

In these approaches, it is important to determine: a) the role of the software system in the organizational context, b) the users of the software system-to-be, and c) the impact of the system on the performance of the organizational processes.

The resulted knowledge will help in building a software system that works harmoniously with the organizational processes.

Goals, a set of objectives the system under consideration should achieve in order to meet stakeholders' needs, play a very important role in this phase; they have been recognized as a basic tool in Requirements Engineering [11]. For this reason, they have been used in the early requirements phase to help in obtaining both the functional and the non-functional requirements for a software system.

A reason for using goals in the early Requirements Engineering phase is that they allow the visualization of states that an enterprise (desires) expects to achieve. Goals also provide the purpose and reasoning that will justify each one of the requirements of the information system.

The next section presents one the main concepts that are used in the early requirements phase: the goal-oriented modeling.

2.3 Goal Modeling

Goal modeling is a prominent formalism that intends to address the early-phase of requirements engineering [11], in which stakeholders, their goals (intentions) are explored and alternative system proposals that satisfy these goals are investigated.

In [11], Lamsweerde defines the goal concept as "a goal is an objective the system under consideration should achieve" and it allows for capturing requirements at different levels of abstraction, from high level, representing strategic concerns, to low level, representing the technical concerns of the system. A remarkable quality is the possibility of recording the rationale behind them (the *why*), complementing the "*what"* and "*how"* dimensions that classical modeling approaches address. In goal-oriented RE, the relationship between the requirements and their motivating goals is represented explicitly, *i.e.* the goal graph provides traceability from strategic concerns to technical details.

Goals can be used for requirements elaboration, verification or conflict management. They are also used to explain the requirements to stakeholders, and the notion of goal refinement provides a natural mechanism for structuring complex requirement documents.

Nowadays, several research efforts use goal mechanisms during the requirements elicitation process.

One of the most relevant works in this field is the KAOS approach [11] [12] [13]. KAOS is a requirement elicitation technique that provides formal rules for analyzing goals and producing requirements based on pre-stated goals. It also provides support for finding alternatives to satisfy the organizational goals. KAOS is a specification language based on concepts such as: object, action, agent, goal, constraint, etc. This language uses real-time temporal logic to represent constraints on past and future states. Hence, the use of this approach is restricted to analysts that are used to deal with formal methods as a current concept in their modeling activities.

This method includes classical questions, such as how and why, to refine and abstract goals in the goal-reduction graph: the identification of pre, post and trigger conditions of goals, the identification of agents to which goals are to be ascribed, identification and resolution of conflicts, etc.

KAOS classifies goals into: achieve, cease, maintain, avoid and optimize goals. Achieve and cease goals are said to generate behaviors. Maintain and avoid goals are said to restrict behaviors. Optimize goals are said to compare behaviors [13]. This classification enables the analyst to capture the complex organizational setting.

Figure 2.1 depicts an example of a portion of a possible goal structure of a borrower in a library system and which is modeled by KAOS in [13].

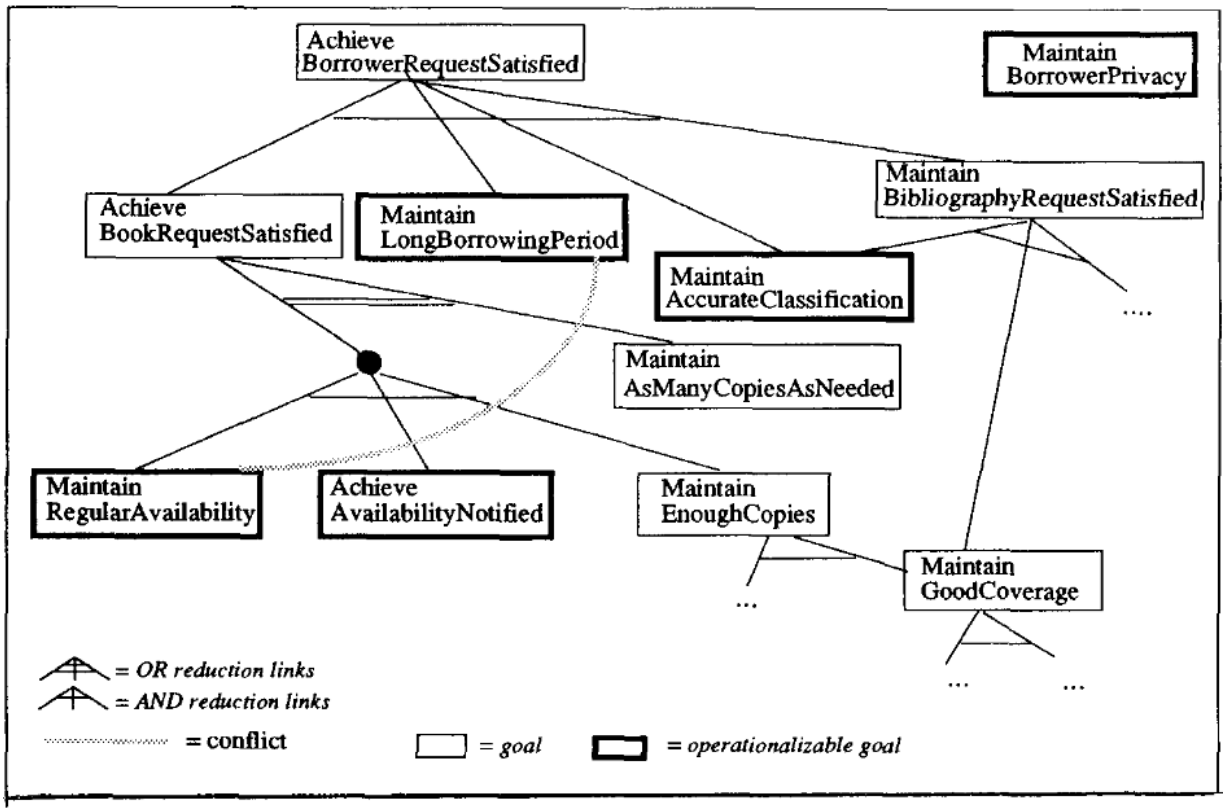


Figure 2.1: An extract of a goal structure for a library system's borrower using KAOS (source: Fig.3, p.30 from [13])

An additional goal-oriented method is GBRAM (Goal-Based Requirements Analysis Method) [14] [15], in which the (high level) goals are used as the appropriate mechanisms to identify and justify the requirements of a software system according to the business model.

In this technique, a bottom-up approach must be followed to elicit the requirements. This is because the goals are obtained from the description of the current processes and also from the descriptions of the stakeholders. GBRAM is composed of two main processes: goal analysis and goal-refinement.

Figure 2.2 visualizes the different modeling activities of the GBRAM method that appears in [15].

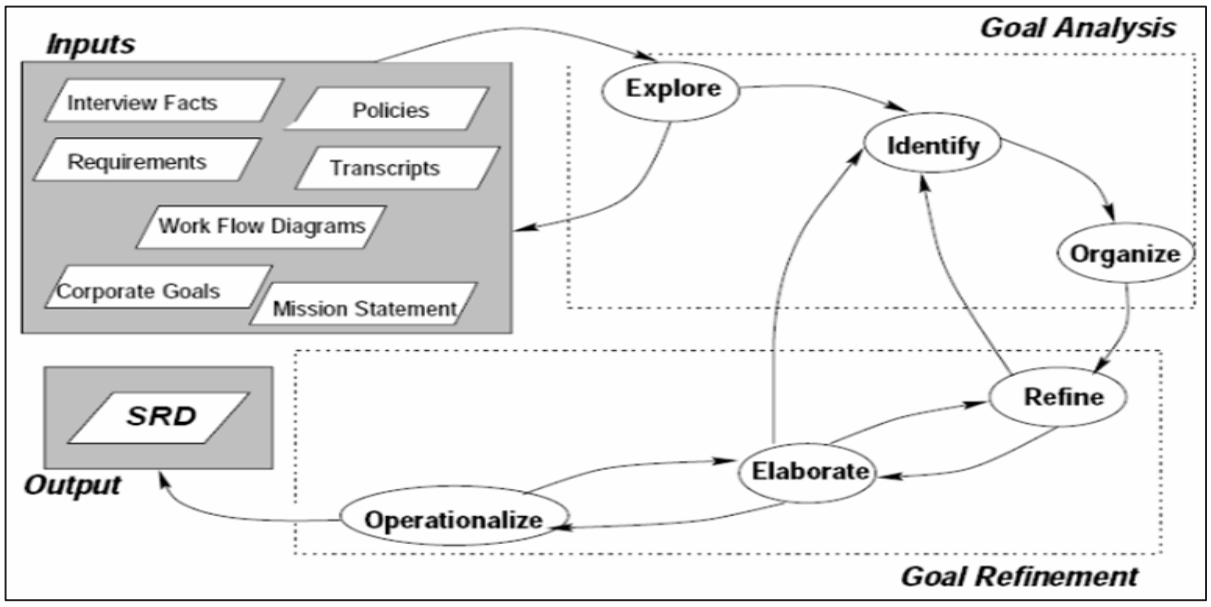


Figure 2.2: GBRAM modeling activities (source: Figure 1, p.158 from [15])

However, this method does not establish a clear distinction between the information used in the early and late requirements phase [10]. Hence, GBRAM does not have a clear representation of the complete process for software development.

Another important research work on goal modeling is the NFR modeling framework [16]. It introduces the goals of the stakeholders explicitly using a graphical notation, and uses these goals to derive system requirements and design activities. The NFR framework uses the notion of softgoals (by definition fuzzy or ambiguous goals) and contribution links.

Softgoals are goals that are not satisfied via clear-cut criteria, and contribution links represent potentially partial negative and positive contributions to such goals. These constructs produced a qualitative framework, able to represent non-functional requirements which are more difficult to define rigorously, such as security, performance, usability and cost. Figure 2.3 illustrates an example of a NFR model that was shown in [16].

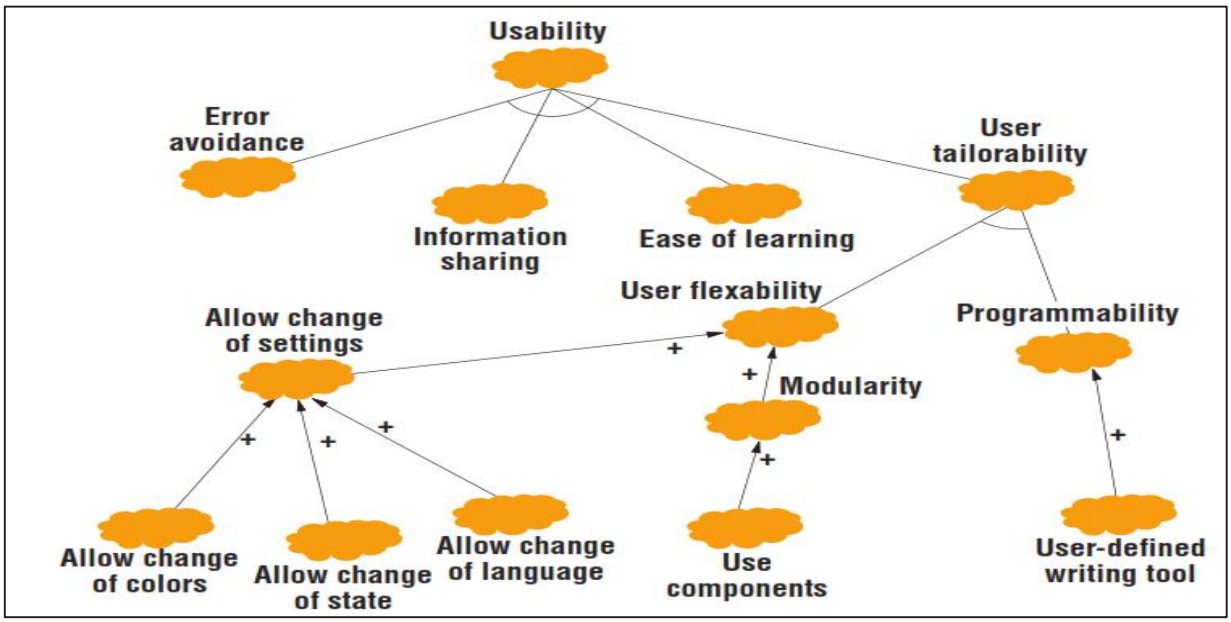


Figure 2.3: An example of a partial hierarchy for the softgoal "usability" using NFR (source: Figure 2, p.94 from [16])

To sum up, this approach focuses on analyzing the impact of non-functional requirements in the software development process.

As i* framework is the main focus of this work, we will devote the next section to investigate and explore it in details.

2.4 The *i** Framework (iStar the 1.0 version)

(Almost all the definitions in this research comply with the ones existing in the *i Wiki [4] and specifically for iStar 1.0)**

The *i** (pronounced i-star where *i* represents the distributed intentionality) is a goal-oriented and agent-oriented framework and modeling language proposed by Eric Yu in his PhD thesis in 1995 [2] with the aim of modelling and reasoning about organizational environments and their Information Systems.

This modeling framework views organizational models as networks of social actors that have freedom of action, and depend on each other to achieve their objectives and goals, carry out their tasks, and obtain needed resources.

The *i** methodology is different from traditional mechanisms to requirements specification that are based on the description of what must be done in order to accomplish an organizational process.

It is well equipped to expose why business processes are executed in a specific way, and also it permits the explicit representation of the space of alternatives that exist for fulfilling a business goal. Furthermore, it permits omitting the operational details of the processes by reducing the complexity of the business model which allows having a high level representation of the current as well as the future enterprise situation.

It has spread and successfully been implemented in different contexts, e.g. Business Process Reengineering, organizational modeling, requirements elicitation, software design and even security. As a result, this broad scope has given rise to an extended practice in the construction and analysis of *i** models (more details can be found in **i* Wiki** [4]).

The *i** modeling language is constituted basically by a set of graphic constructs which can be used in two types of diagrams.

A metamodel for the *i** modeling language (Figure 2.4), was earlier proposed and revised in [30], describes the abstract syntax of the language by means of meta-classes, meta-associations and cardinality constraints. It can serve as a precise description of the notation and is therefore useful in implementing modeling tools, since it can be used as a basis to define the language syntax.

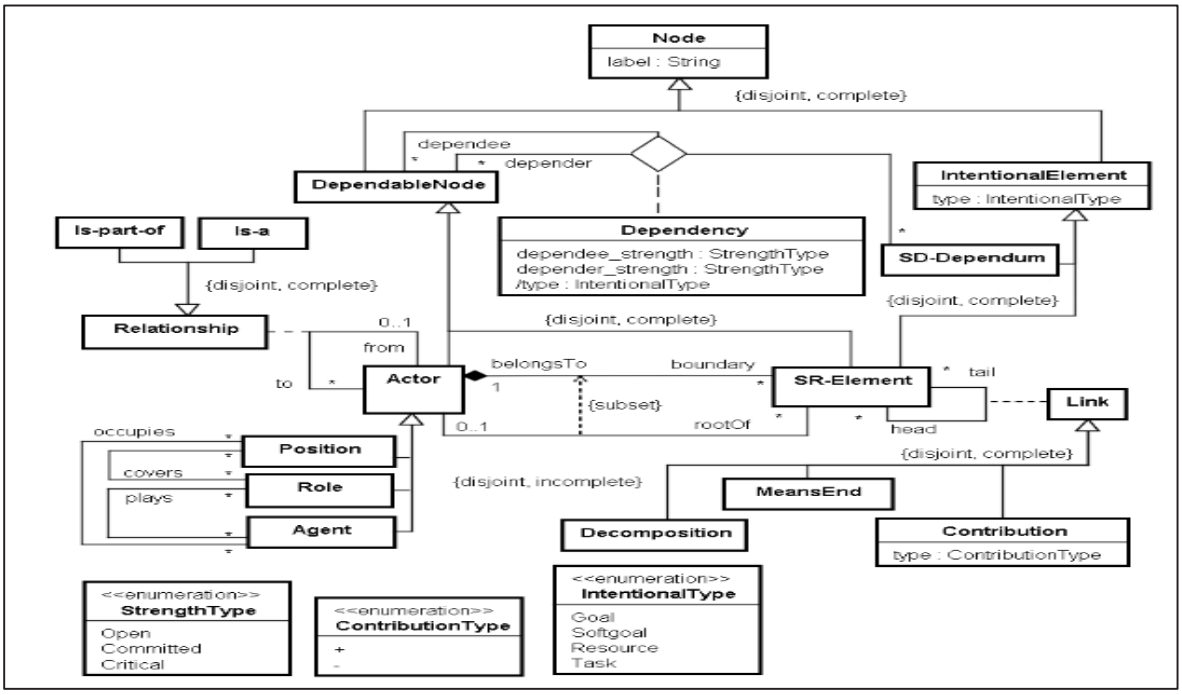


Figure 2.4: The *i revised Metamodel (taken from [30])**

2.4.1 Overview of the *i** Models

*i** is characterized by defining two levels of diagrams which are used to describe system requirements: 1) the strategic dependency (SD) model for showing how actors/agents in a system depend on one another and 2) the strategic rationale (SR) model for expressing the goals of each actor/agent and how each goal can be reasoned and achieved.

2.4.1.1 The Strategic Dependency (SD) Model

It is a graph involving a network of nodes and links, as it is shown in Figure 2.5, where each node represents an actor and each link maps out one dependency between exactly two actors. It depicts the strategic dependencies between actors.

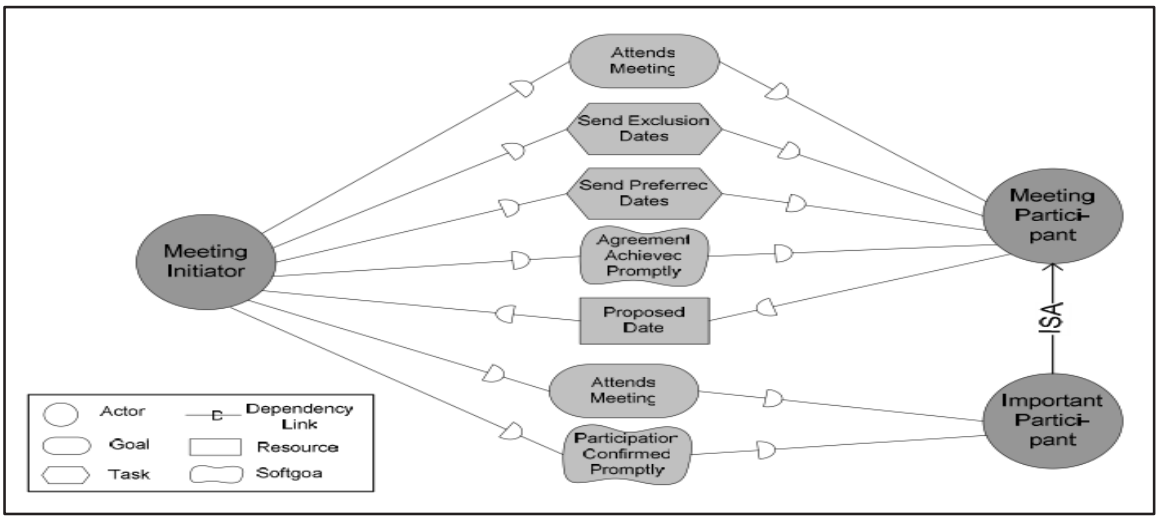


Figure 2.5: The meeting scheduler example [2] (SD model)

The elements that characterize this model are stated one by one below:

Actor: is the central concept in *i** and it represents an entity that has strategic goals and intentionality within the system or the organizational setting.

The actor is an active entity (can be human, abstract, or electronic) that carries out actions to achieve goals by exercising its know-how. So, organizational actor is viewed as having intentional properties such as goals, beliefs, abilities, and commitments.

The term actor is used to refer generically to any unit to which intentional dependencies can be ascribed. By depending on others, actors may be able to achieve goals that are difficult or impossible to achieve on their own and, consequently, they become vulnerable if the depended-on actors do not deliver what the former ones want and need. Actors are strategic in the sense that they are concerned about opportunities and vulnerabilities, and seek the arrangements of their environments that would better serve their interests.

Actors are represented graphically as a circle containing the actor name. The actor notion can be expanded into the more specific constructs of an *agent*, *role*, or *position* (see Figure 2.6, left side).

Agents represent particular instances of people, machines or software within the organization and they occupy positions and, as a consequence, they play the roles covered by these positions (see Figure 2.6, right side).

Actor association Links: The relationships between actors are described by graphical association links between actors. The types of actor association links are:

- Is part of: it is used when an actor is part of another actor. Roles, position and agents each can have subparts.
- Is a: it is used to represent a generalization with an actor being a specialized case of another actor.
- Plays: it is used between an Agent and a Role, with an Agent playing a Role.
- Covers: it is used to describe the relationship between a Position and the Roles that it covers.
- Occupies: it is used to show that an Agent occupies a Position, meaning that the Agent plays all of the roles that are covered by the Position.
- Instance of: it is used to represent a specific instance of a more general entity. An agent is an instantiation of another Agent.

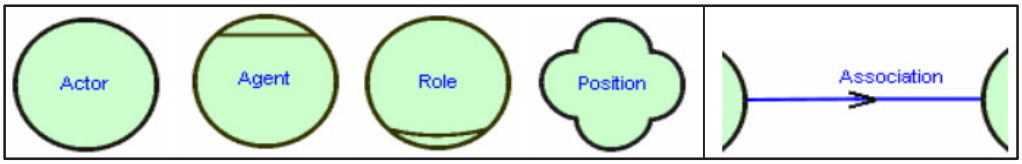


Figure 2.6: Graphical Notations of Actor, Agent, Role, Position and Association

Dependency: A dependency is an explicit "intentional" relationship (link) among actors which expresses that an actor (**depender**) depends on some other actor (**dependee**) in order to obtain some objective (**dependum**). The dependum is an intentional element that can be a resource, task, goal or softgoal [2]. The types of strategic dependencies, based on the type of the dependum are:

✧ **Goal Dependency.** In goal dependency the *depender* depends on the *dependee* to bring about a certain state of affairs in the world. In goal dependency, all decisions about fulfilling the goal need to be taken by the *dependee*, therefore, the *depender* doesn't care about how the *dependee* goes about achieving the goal. Put differently, the *depender* delegates the responsibility for fulfilling the goal to the *dependee*, who is the new goal owner.

✧ **Softgoal Dependency.** The *depender* depends on the *dependee* to meet some non-

functional requirement. A softgoal is similar to a goal except that the criteria of success are not sharply defined a priori. The meaning of the softgoal is elaborated in terms of the methods that are chosen in the course of pursuing the goal. The *depender* decides what constitutes satisfactory attainment of the goal, but does so with the benefit of the *dependee* know how.

✧ **Task Dependency.** In task dependency a *depender* depends on the *dependee* to execute a given activity (accomplish some specific task). The *depender* is the actor that prescribes (imposes) the procedure to execute the delegated task, in this sense; the *dependee* has already made decisions about how the task needs to be carried out.

✧ **Resource Dependency.** In a *resource dependency*, the depender is depending on the provision of some entity, physical or informational. This type of dependency assumes no open issues or questions between the depender and the dependee.

The graphical representation of each type of dependency is shown in Figure 2.7 below.

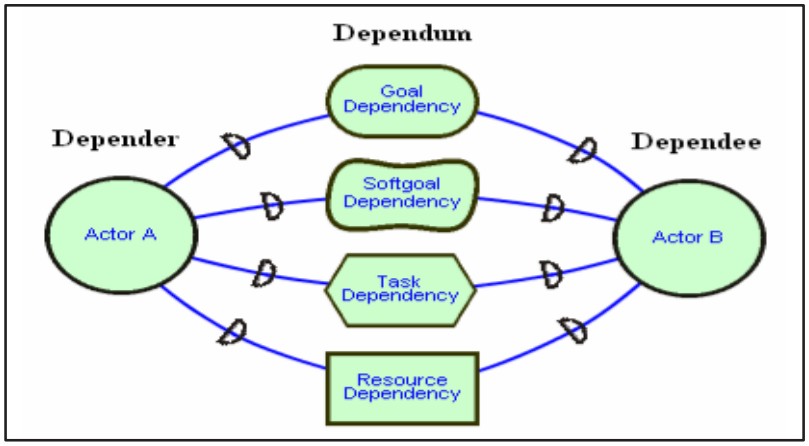


Figure 2.7: Graphical Notations of Dependencies Types

2.4.1.2 The Strategic Rationale (SR) Model

Strategic Rationale (SR) model is a graph, with several types of nodes and links that work together to provide a representational structure for expressing the rationales behind dependencies. SR diagrams (see Figure 2.8) open up actors and show all the internal elements, including goals, softgoals, tasks and resources, besides three types of internal links inside the *i** actor (means–end link, task-decomposition links and contribution links). These elements are described below:

Actor Boundary: An actor boundary indicates intentional boundaries of a particular actor. All of the elements within a boundary for an actor are explicitly desired by that actor. In order to achieve these elements, often an actor must depend on the intentions of other actors, represented by dependency links across actor boundaries.

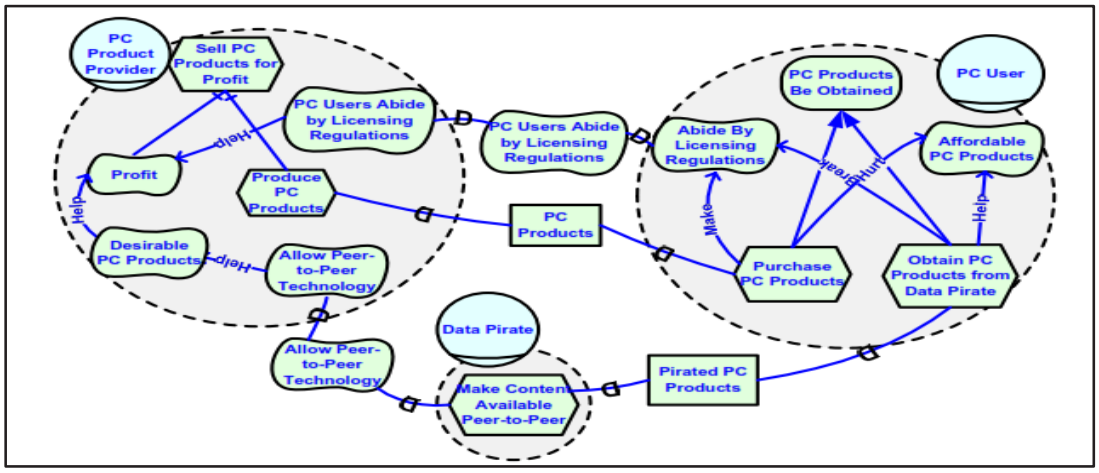


Figure 2.8: Example of SR Model "from the Trusted Computed Case Study (Soure: Figure 1.1, p. 16 from [18])**"**

Intentional elements: An intentional element is an entity which allows to relate different actors conforming a social network or, also, to express the internal rationality of an actor. The types of intentional elements are:

- **Goal:** it represents a condition, intentional desire or state of concerns that the actor would like to obtain (achieve).

- **Softgoal:** it is similar to the goal except that the criteria for the goal satisfaction are not clear-cut. It is judged to be sufficiently satisfied from the point of view of the actor. The means to satisfy the softgoals are described using contribution links from the other modeling elements.

- **Task:** it specifies a particular way of doing something in order to fulfill some goal.

- **Resource:** is a physical or informational entity (a means) that must be available for an actor to perform some task.

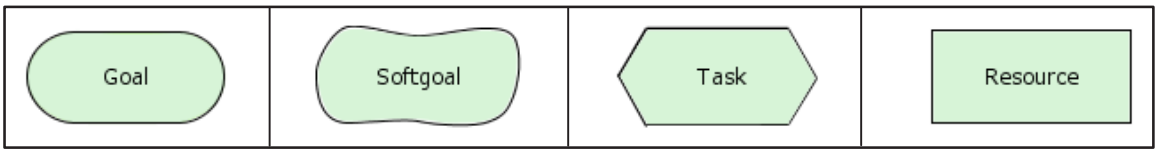


Figure 2.9: Graphical symbol of different Intentional Elements

Intentional element relationships: An intentional element link represents an n-ary (when n is included in [1..n]) relationship among intentional elements. The types of intentional element relationship are:

- ***Means-Ends (ME) links:*** These links indicate a relationship between an end, and a means for attaining it. The "*means*" is expressed by using the concept of task, since the notion of task embodies how to do something, with the "*end*" is always expressed as a goal. Accordingly, ME links may break down a goal into alternative tasks that achieve the goal or to just one possible way to fulfill it (cardinality is [1..n]).

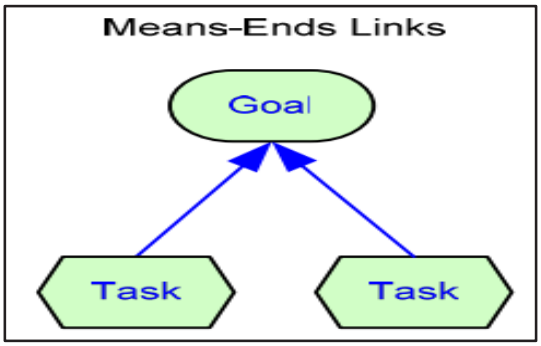


Figure 2.10: Graphical Notation of Means-End links

- ***Task-Decomposition links*:** A task element is linked to its component nodes by decomposition links (AND Decomposition). A task can be decomposed into four types of elements: a subgoal, a subtask, a resource, and/or a softgoal. The task can be decomposed into one to many of these elements.

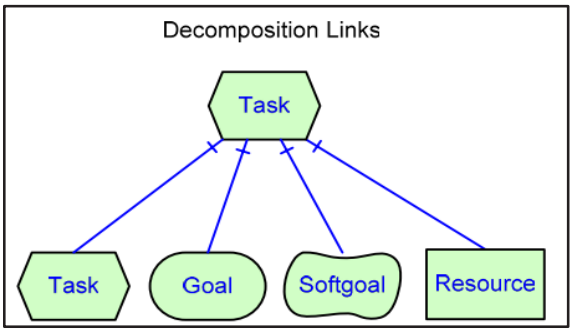


Figure 2.11: Graphical Notation of Task-Decomposition links

- **Contribution links:** The contribution links are: make, some+, Help, Break, some-, hurt, unknown, AND, and OR. These contribution links can be used to link any of the elements to a softgoal and contributes to the satisfaction or fulfillment of the softgoal *i.e.* they express the impact of an element on softgoals.

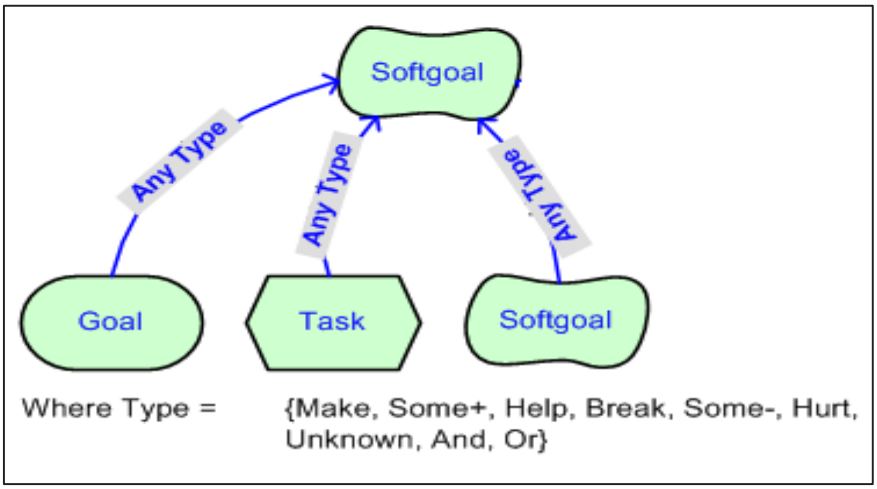


Figure 2.12: Graphical Notation of Contribution links

2.4.1.3 Concepts not covered (out of this thesis scope)

The context of the present work covers the *i** (iStar) framework and modeling language and precisely the iStar on its 1.0 version. As mentioned earlier in this work, the *i** modeling language offers a set of graphical concepts that are used for constructing the *i** requirements models both SD and SR. In this research work, we focus on a subset of the *i** graphical notation elements *i.e.* the common and widely used basic concepts when sketching the *i** diagrams. Thus, there are some few modeling constructs which are not under the scope of (not covered by) our work such as the belief element, the actor specific constructs (agent, position and role) and the actor association links (is part of, is a, plays, covers, occupies and instance of).

2.5 Summary

This chapter introduced the main concepts and notations necessary for the remainder of the thesis. An overview of the Requirements Engineering phase, early RE and goal modeling has been presented. The *i** framework with its models and their main primitives were described in details here.

Chapter 3. Supporting the iStar Model Quality Review

This chapter intends to present an overview and revision of some freely available *i** modeling language editors and tools. We mainly assessed and tested their syntactical checking features in order to tell the reasons that make them insufficient and inadequate to use in a teaching/learning *i.e.* an educational context. The results that we got from closely studying these tools were the relevant triggers for the conception of our proposed research work. This chapter focuses on the first problem treated in this thesis. It unwinds our proposal to solve it by describing a web application, called *i**Check, which was proposed and developed to complement and enrich the aforementioned available tools that turned out to have limited syntax and model quality checking features. Also, this chapter includes reports on the experiments that we conducted in order to evaluate our proposed solution. On Section 3.5, we present some observations and findings concerning mainly the issues that we encountered when dealing specifically with the development of the SR model's checking part. Section 3.6, which is the last section, it summarizes and wraps up this chapter.

3.1 Overview of a selected set from the freely existing *i** modeling tools

Like any other existent modeling technique, the *i** goal modeling language is consolidated by free computer-assisted systems that offer a number of functionalities and features to help users sketch and create their requirements models as well as check their quality. As a first step, we searched and got some of these tools which are mainly state-of-the-art research prototypes. We used them to build both correct and defected (erroneous) models in order to later assess their abilities in catching and detecting any modeling deficits that we deliberately utilized. Our survey of these tools exposed several shortcomings particularly in their syntax checking features' side. It led us to figure out the rising necessity for

complementing these tools' syntactical checking functions. Thus, it motivated the construction of the first tool that we built for this dissertation.

3.1.1 H*i*ME (Hierarchical *i** Model Editor)

H*i*ME [5] is a free *i** tool which does not represent the *i** models graphically through the language symbology *i.e.* visual concepts, instead, it shows them as a folder-tree directory in a file system. It is an editor that includes specific features for dealing with inheritance operations when inheritance appears in the models (**this is out of the interest and scope of this research work**).

This tool uses iStarML [19] as the unique format for storing the created *i** models.

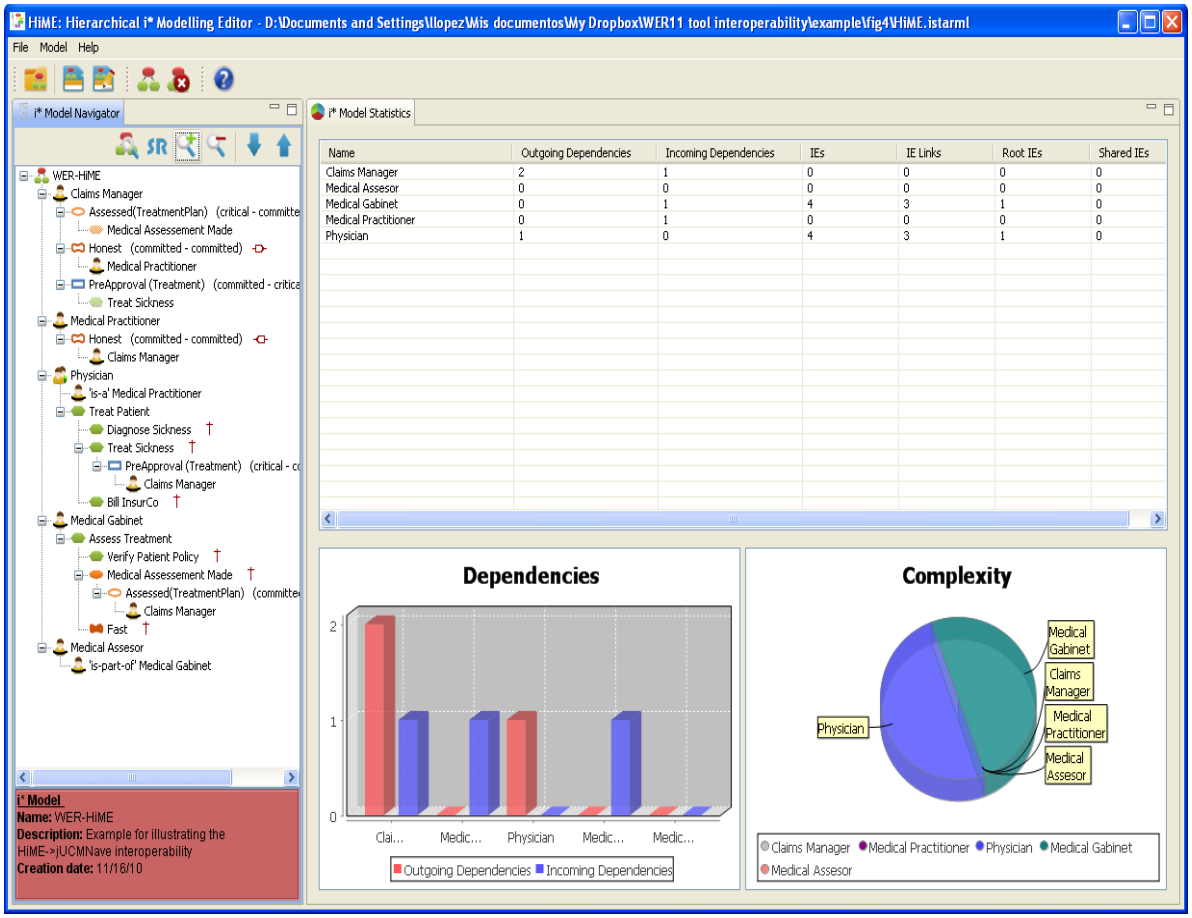
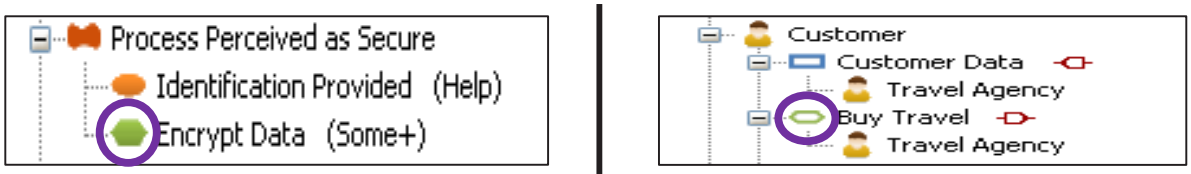


Figure 3.1: General view (main window) of the H*i*ME tool (source: Fig.1, p.4 from [36])

As it is shown in Figure 3.1 above, HiME's main window is divided in two parts. The model navigator (left side) is used for viewing and managing the model. The icons that appear in this view give some

information about what kind of element is represented. Now, the model statistics (right side) shows information about the complexity of the model.

To our experience, H*i*ME requires an effort and practical training time in order to get used to it because the model readability is not intuitive neither straightforward to it new users. For example (see Figure 3.2), as it is mentioned in the H*i*ME user guide [36], the meaning of icons, when a task element is a full shape, this means that this task element is an internal element *i.e.* an SR element which is used only inside the boundary of an actor. Whereas, when a task element is just an "empty" hexagon icon, it means that this task is an external element *i.e.* a dependum node from a dependency relationship.



(a) (b)

Figure 3.2: An example showing the iconic filling difference between SR internal elements (a) and external elements aka dependencies nodes (b)

3.1.2 OpenOME: an Open-source Goal and Agent-Oriented Model Drawing and Analysis Tool

The development of OpenOME [7] was initiated in 2004 at the University of Toronto, Canada. It is an Eclipse-based open-source tool supporting the construction and analysis of the *i** models. The tool is in a stable state and available freely for download. It includes support for forward and backward interactive, qualitative *i** analysis and more importantly to us the syntax checking feature.

The tool allows users to graphically draw models using a palette of shapes. Standard features such as saving, zoom, cut, copy, and paste are provided. Models are grouped under user-created projects, shown in a folder view. OpenOME imports and exports models in the GMF .ood and .oom format, as well the iStarML format [19]. See Figure 3.3 for a screenshot of the OpenOME interface.

OpenOME architecture takes advantage of the Eclipse package development, allowing for extension or customization with the addition of a new development package. The current version is 3.4.1 (as of 2011).

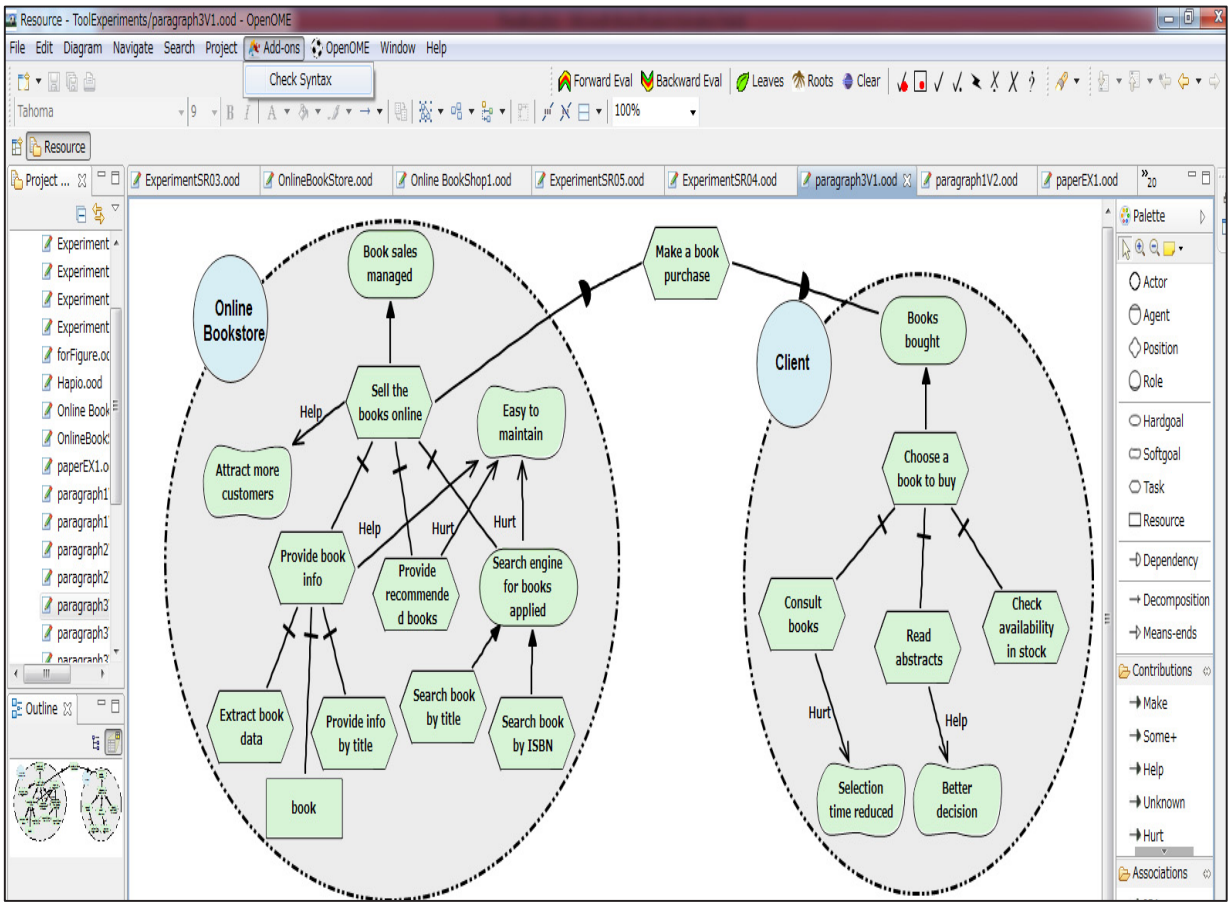


Figure 3.3: A screenshot of the OpenOME tool

3.1.3 iStarTool

The iStarTool [6] supports the graphical modeling of the *i** methodology. It has been developed using the open-source Eclipse platform and model-driven technologies, such as the Graphical Modeling Framework (GMF).The main purpose of the iStarTool is to facilitate the learning of *i** language and improve the quality of *i** models, being especially aimed at beginners.

Unlike the Syntax Warning System, the Syntax Checker runs offline. The reason is that some *i** steps could be considered wrong if analyzed in real time. In order to execute it, the user has only to click on the menu button that is called Syntax checker.

The iStarTool also allows users to work on multiple models at the same time. It saves all models in XML format, by default, and if necessary, they can be exported as an image. Moreover, the current features of the iStarTool can be reused to support families of graphical editors for *i** based languages. Figure 3.4 illustrates the main window of iStarTool.

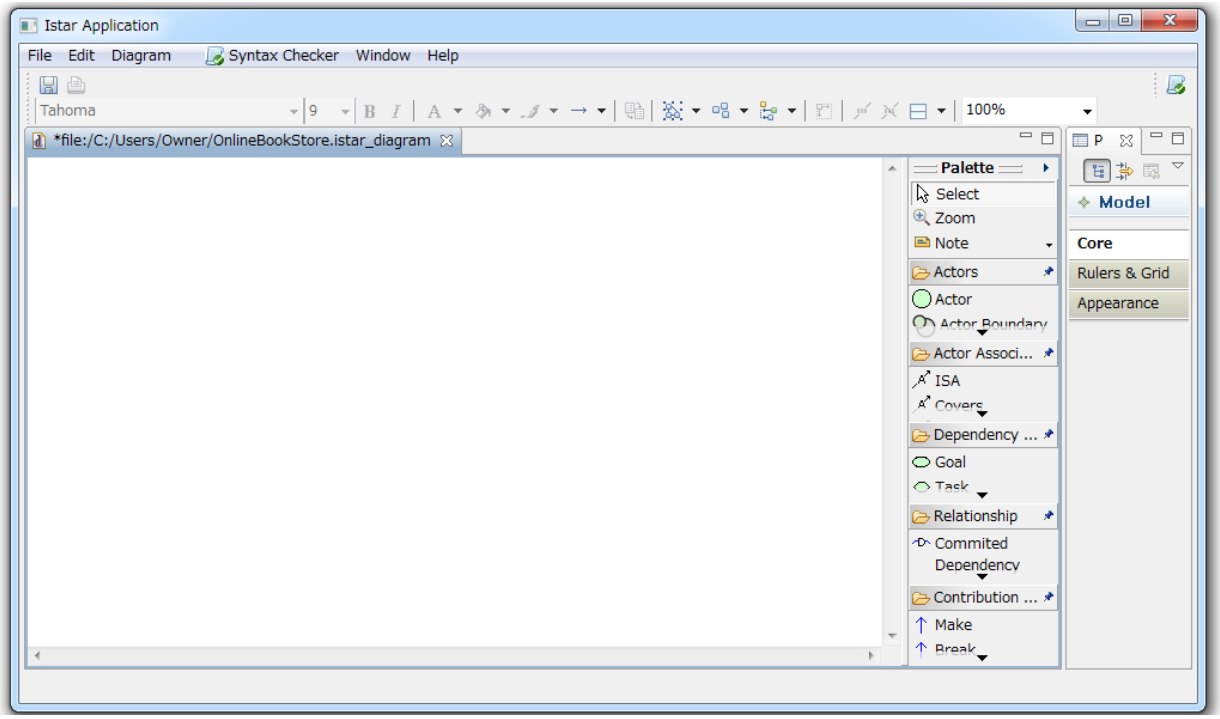


Figure 3.4: Screenshot of the iStarTool interface

3.2 Why another model checker is needed?

*i** is a well-documented goal-oriented modelling language which is used in a variety of fields ranging from RE to business process re-engineering to social modelling. Moreover, it is taught in many universities around the world for more than a decade. However, some applications of even the basic concepts are not straightforward for the novice (*e.g.*, how to determine which of the modelling constructs should be used to describe a particular system requirement, how to properly decompose complex or hierarchical concepts in the requirements description, etc.) and new learners will require some practical training to master the techniques. To do this in a classroom setting, it is desirable to have a computer-aided modeling tool that will make it possible for many students to build and check diagrams on their own at the same time, since reviewing students work and coaching them on how to find errors on an individual basis is time- consuming and arduous. Several free *i** tools are available [5][6][7][8] to help people sketch and draw their requirements models according to the prescribed modeling rules of the *i** modelling language. But, according to a survey that we performed on these tools, none of them implemented nor provided the full and complete set of syntactical quality checking features and this gives novice users plenty of opportunities to build structures containing errors and amenable to originate confusion and misinterpretation without even realizing it. Furthermore, unlearning such bad modeling practices will

preoccupy them as it would cost them effort and time. To put it more simply, by surveying a set of free *i** tools, we gathered evidences of their limitations concerning the model checking features. These tools do not provide an accurate and in-depth checking and this may hinder the production of models with high or even good quality. And clearly, if these tools are used in a classroom setting, with large groups of students for example, it is possible for these learners to construct models containing errors here and there and not realize it at all. Learning the basic concepts of a framework incorrectly at the start can lead to high unlearning costs for handling the correction of bad modeling habits later.

3.2.1 Examples of checking and feedback weaknesses of the surveyed tools

3.2.1.1 The H*i*ME case

Considering the model checking part, H*i*ME does not offer an explicit function (button). Instead, it does force its users to select only its offered concepts. So, by this, it prevents errors occurrence as possible as it can. However, there are some checking gaps, which proves that HiME doesn't offer the full set of checking features and this may hamper the correct learning of the *i** framework, its modeling rules and good habits and practices. Figure 3.5 shows an example of Means-End relationship, which MUST be a relation (link) between a task "the Means" and a goal "the END". Put it more simply, according to [4], a goal can only be refined by the Means-End links. The Means-End Link is a type of a relationship that indicates an End (Goal) and it's Means (Task), or how to achieve the Goal.

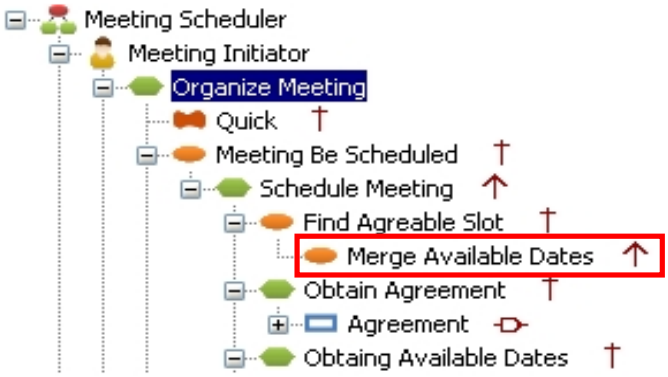


Figure 3.5: An example showing a goal depicted as a Means "Merge Available Dates" in order to achieve a high-level goal which is an End "Find Agreeable Slot" (screenshot was taken from [5])

3.2.1.2 The OpenOME case

As our main focus and aim is to survey and investigate the checking features of the current free *i** tools, we did study the syntax checking features provided by OpenOME and we found out that it has limited and incomplete set of *i** framework syntax checks. Also, sometimes even if it does detect the error, it delivers

error or warning messages which are not straightforward and intuitive to the modeler. So, they can lead to confusion and may require additional effort in order to correct the caught modeling defect. Figure 3.6 shows some examples of returned errors and warnings from the syntax checker add-on of OpenOME.

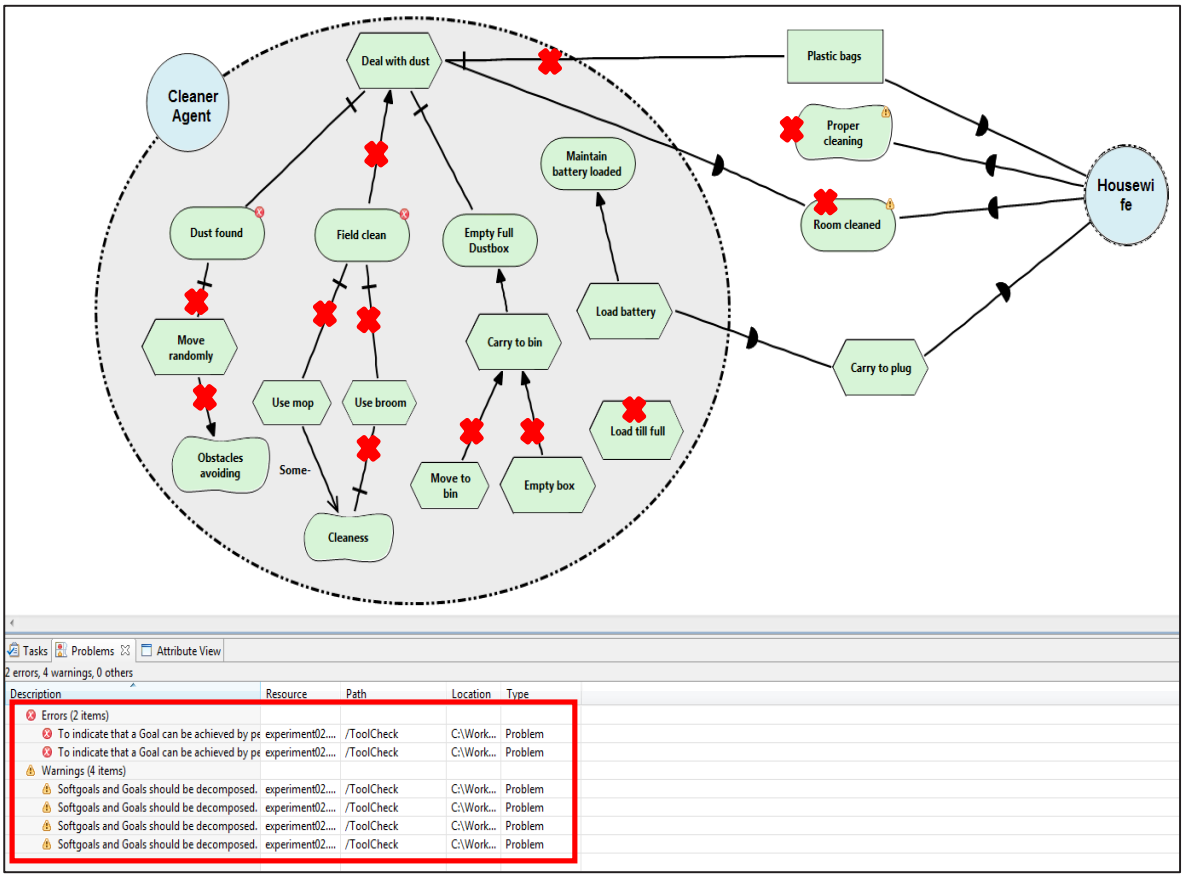


Figure 3.6 Screenshot of the returned checking feedback offered by the OpenOME's checker

To start with, the model appearing in Figure 3.6 above does contain 12 defects (if we count the number of ✖ occurrences that it is deliberately annotated by this thesis author). However, the checker returns a feedback summary which only lists two errors and four warnings that are being of the same type, at least this is what the message suggests, each. To sum up, the checker, not only its feedback is limited but also sometimes, the returned messages are unclear and that may cause confusion to the user. A positive point that OpenOME checker provides is that it pinpoints and locates where in the model the defect appears (it specify the certain and particular element that causes the issue or at least it is involved in some defect).

3.2.1.3 The iStarTool case

Syntax errors are one of the biggest problems for students learning a modeling language as they slow students' progress. Typically, error messages are an important tool for novice modelers to use as they help

them locate and fix mistakes or issues in their built diagrams. When an error message is unhelpful, it can be difficult to find the issue and may impose additional challenges in learning the language and its concepts. To this respect, another body of work has focused on the development of a tool called iStarTool. It allows students to create *i** goal models and also it provides automated help to them consisting in the syntax checker feature which returns a feedback comprising a list of error messages. However, this tool's returned feedback is frequently incomplete and inadequate. For example, Figure 3.7 encompasses seven errors that exist in the diagram; however, the syntactical checker returns a list that comprises only three errors. And as we can see, the error messages contents are not beneficial for novice learners since they may add confusion and misunderstanding that may lead the learners down to the wrong path, as well as, it is possible that they introduce some new errors.

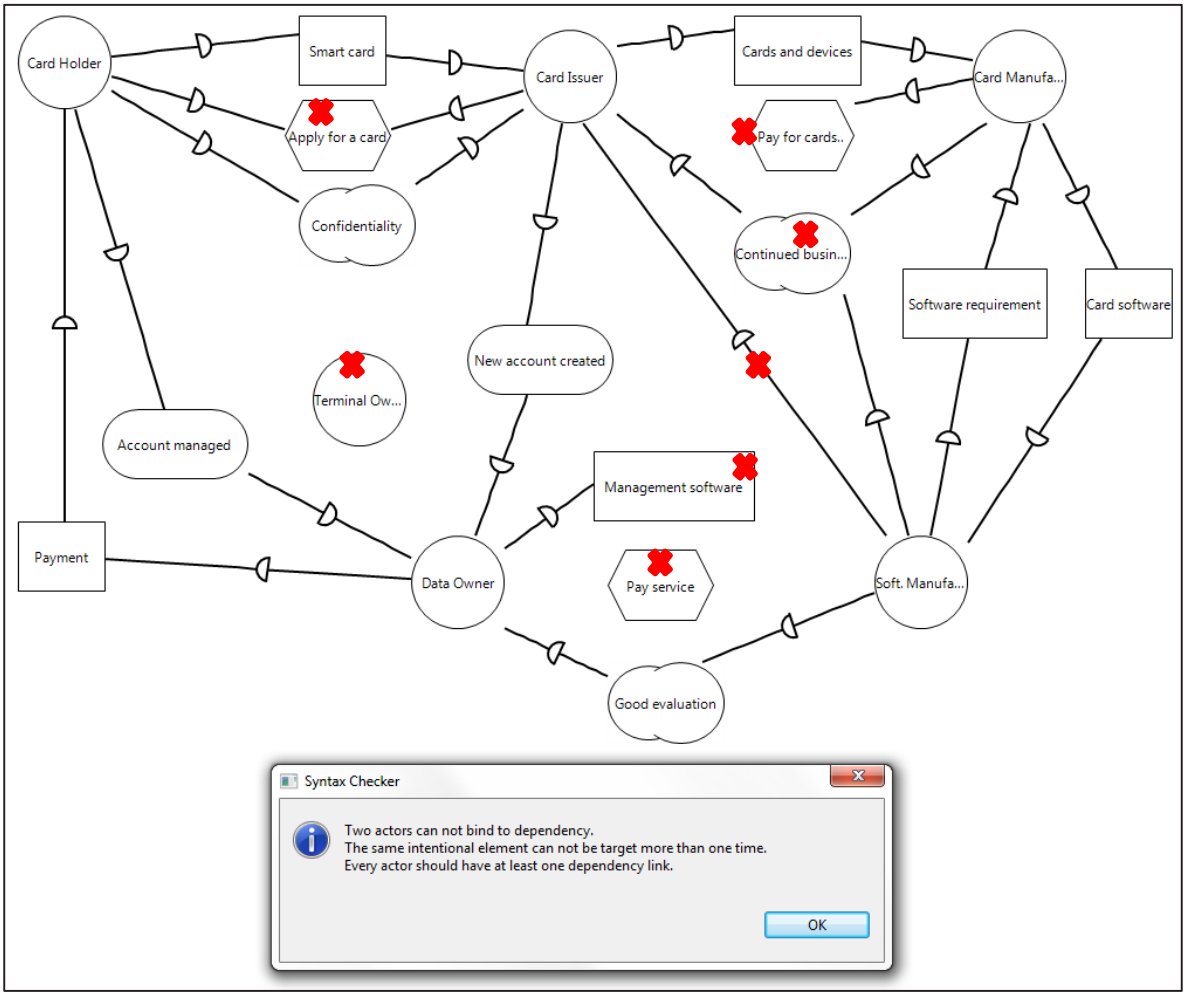


Figure 3.7: A screenshot of the returned checking feedback offered by iStarTool syntax checker

Let us take the returned message “The same intentional element cannot be target more than one time”. For example, to our understanding, it refers to two distinct construction rule violations namely number 4 and number 9 in Table 3.2 (see sub-section 3.3.3.1), but it can be confusing to a user, particularly a new user to the framework, because 1) the meaning of the message is not clear and concise and 2) this message is addressing two different defects. And, since only one message entry is displayed to the user, the later can be confused to choose the right defect location in order to introduce the necessary correction.

To cover this gap in the current *i** tools landscape, we developed a web-based system called *i**Check to complement the checking features of the available tools by offering the users automatic on-demand feedback on the quality of their diagrams *i.e.* allowing them to review and check their models to subsequently introduce the necessary correction and improvements as soon as possible to their build-time.

3.3 *i**Check: Description and Evaluation

3.3.1 Approach Overview

Given the necessity and the need for an in-depth model checking according to the *i** language rules and conventions and also in order to achieve a good quality artifact, we built a web-based tool, *i**Check (requiring no download or installation of software), which aims: 1) to help novice modelers recognize their errors that result from the misinterpreted use of the *i** modeling language constructs and 2) to offer recommendations on how to eliminate these errors by using textual and visual feedback and information. Particularly, for learners in a classroom setting where an instructor might not always be available to point out errors in individual graphical representations, the *i**Check system will be a useful way for students to obtain a thorough automatic feedback about the quality of their *i** depictions of system requirements. The rule violations detected by *i**Check are indicated to the user through a browser interface along with specific tips and GIF animations correction scenarios on how to resolve each specific problem.

Various *i** tools use their own file format for saving model data. But some of these tools generate a text-based extensible markup language (XML) format called iStarML [19] that can be used to export the model data to other systems. As its usage is outlined in Figure 3.8, the *i*Check* application accepts this produced iStarML format data as input and evaluates the quality of the model using a checklist of the *i** core rules. A list of rule violations detected by the system is indicated to the user along with specific tips and examples on how to resolve each problem. By providing a means for learners to obtain feedback about the quality of their *i** depictions of system requirements in real-time, we let beginners-to-*i** learn how to identify and correct their mistakes as they practice the various design techniques and construct diagrams of higher quality without the need for constant one-on-one human coaching. Since it is hard for an instructor, because it is time-consuming and error-prone, to be constantly available to closely

superintend students' work and offer them guidance and advice on the proper usage of this particular RE methodology throughout their learning process. In the same way, the development and the existence of such tool will bring benefits to the self-studying people.

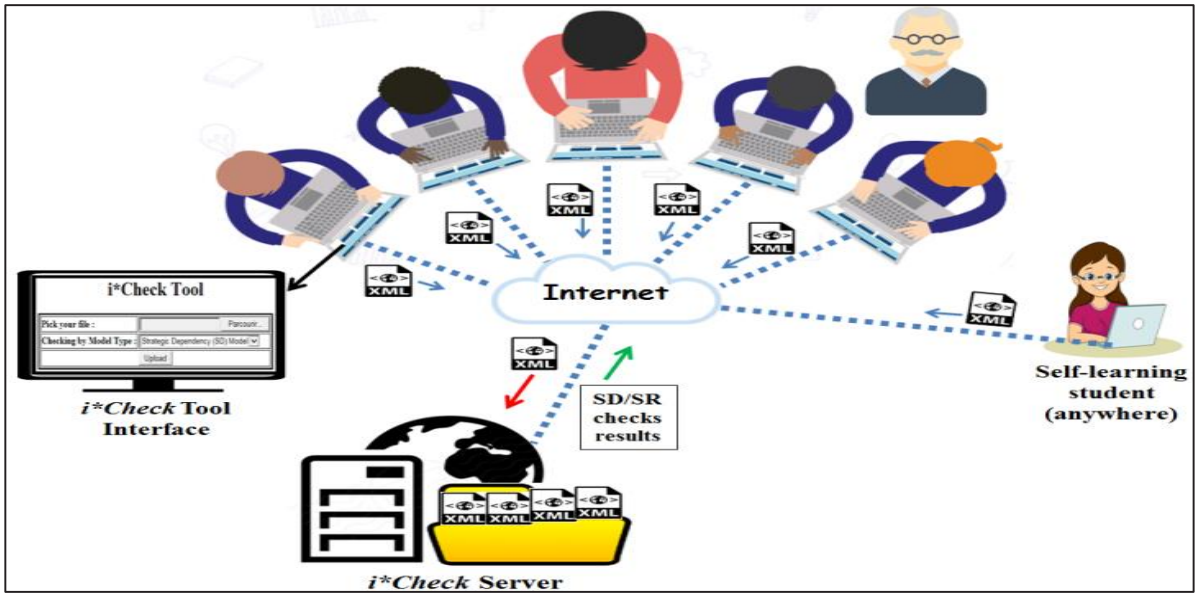


Figure 3.8: *i*Check* tool, whether used in a classroom setting or elsewhere, it evaluates the Strategic Dependency (SD) and Strategic Rationale (SR) models and indicates the syntax problems to be fixed and the animated steps for a particular defect correction.

We implemented the *i*Check* application using the Hypertext Preprocessor (PHP) [20][21] which contains a library providing Document Object Model (DOM) [21] XML functions that enabled us to parse, select particular elements and their attributes from iStarML files and subsequently work on them.

The animations were first created as slides in Microsoft PowerPoint 2010 and then the created slides (files) were uploaded to a free website that transformed them to GIF animations (see Figure 3.9). They aim to offer step by step hints and correction suggestions to the novice learners (those who are inexperienced as they are taking introductory modeling courses) in case of detected model defects. The animations are displayed (generated by JavaScript which makes them expanded "showed" and collapsed "hidden" according to the user's will) to the users when they click on a link that will load the animation on the same feedback results' page in order to not to distract their attention with different pages.

There is a variety of reasons why we decided to develop *i*Check as a web-based application. To start with, web applications are not tied to a single computing environment. Plus, no installation is required besides having a browser which students generally have on their PCs. Finally, when the tool is available

online, anyone (student or even expert) can access it anywhere (classroom, laboratory or at home) where there is an internet connection and use it with total freedom.

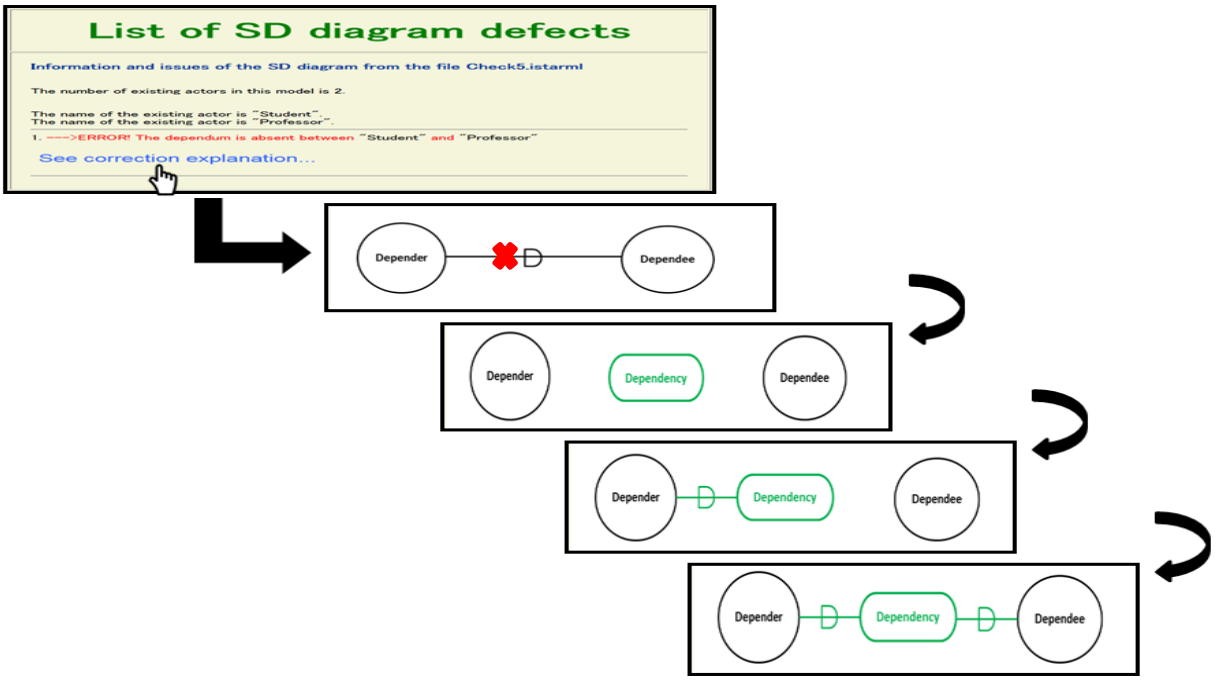


Figure 3.9: In addition to a textual description of each error detected, the *i*Check* tool offers suggestions on how to fix the error in short animations.

3.3.1.1 Feedback improvement

Error messages are particularly critical for introductory and novice modelers in understanding problems appearing in their diagrams. In this context, our *i**Check tool intends to improve novices' debugging performance by providing enhanced error messages that include a more verbose description of the defect than the ones that are provided by current existing systems.

Besides, our enhanced feedback system displays for each defect type the corresponding correction suggestions (sometimes, an error has more than one alternative that can eliminate and thus correct it) detailing how the detected mistake could be corrected. The GIF animation contains an error example and states how it can be corrected (some brief explanation consisting in a list of potential corrective actions that can give extra assistance to the novices). Either, the correction-oriented GIF animation provides one or more alternatives for solving a defect; it is up to the novice learner to choose and decide the one way that makes sense to him *i.e.* the one that corresponds to his conception and his modeling intention. It is important to mention that the GIF animation does repeat, from its beginning, as many times as wished by the students and this can be beneficial for them as we believe that they need multiple repetitions of information before it is understood and memorized.

To put it more clearly, *i**Check doesn't force the beginner to select a specific solution. On the contrary, it does provide him with a set of solution scenarios so that the beginner can choose one specific desired solution or rethink his modeling choice. Furthermore, *i**Check is not like the Microsoft Word automatic spelling checker and corrector neither it is like any programming language compiler which guarantees that only correct code is required to proceed the performance of the desired behavior of the program. In other words, *i**Check "does not do the work for the student". Instead, it does help its users to recognize, locate and detect the defects that may reside in their models and each defect type is attached with one alternative or more proposing how to introduce and thus rectify the error. The final decision is left to beginner as he is the responsible of the revision of his modeling style and choice (he has to rely on his understanding and judgement and then proceed the correction work with the more convenient solution according to him) and this is what proves that the use of *i**Check is completely voluntary.

3.3.2 iStarML: Definition and Basic Structure

As mentioned before, a family of variants was developed based on the concepts of the original *i** framework and modeling technique. The *i** family includes the Tropos methodology [31, 32], the extension of Tropos [33] and Goal Requirement Language (GRL) [34].

As it can be expected, each variant was consolidated with the building of its corresponding tooling supports. From these points emerged the necessity of having a common format for enabling interoperability among *i** family tools. For this purpose, iStarML [19], an XML-based format which focuses on supporting data interchange among different *i** variants' tools, was developed.

In other words, a common representation allows i) to have an interchange format among *i** variants, ii) the representation of differences and similarities among variants and, iii) to have a repository common of *i** concepts.

The most important features of iStarML format is that the different *i** variants can eventually be translated into iStarML [19]. Therefore iStarML allows a textual representation of domain models, requirements, actor relationships and a wide set of the different uses that *i** has covered as modeling language.

The tag <istarml> is the main tag in iStarML. It can contain only the <diagram> tag. In figure 3.10, the options of this tag are shown. Under this structure, it is possible to store on the same file a set of different *i** diagrams. The derivation of iStarML tags from the *i** core concepts has permitted keeping the language simple and, at the same time, to consider different language variations using the same language constructs.

istarmlFile ::=	<istarml version="1.0"> *diagramTag* {*diagramTag*} </istarml>
diagramTag ::=	<diagram *basicAtts* [author=*string*] {*extraAtt*} > [*graphic-diagram*] { [*actorTag*] | [*ielementExTag*]} </diagram>
extraAtt ::=	*attributeName* = *attributeValue*
basicAtts ::=	[id="*string*"] name="*string*" | id="*string*" [name=" *string*"]

Figure 3.10: the <iStarML> syntax

The table below (Table 3.1) summarizes the basic core concepts of the *i** family of variants and their corresponding suggested XML tags which are specified using iStarML.

Table 3.1 Core concepts of *i-based modeling languages and proposed XML tags for iStarML**

*i** core concept	iStarML Tag	Main attributes or subtags
Actor	<actor>	*type* attribute to specify different types of actors (e.g. agent)
Intentional element	<ielement>	*type* attribute to specify different kind of intentional elements (e.g. goal)
Dependency	<dependency>	Can contains two subtags: <dependee> and <depender>
Boundary	<boundary>	*type* attribute for representing future variations on boundary conceptualizations
Intentional element link	<ielementLink>	*type* attribute to specify types of intentional relationships (e.g. contribution) *value* attribute to specify values related to the relationship (e.g. +,++,-,++)
Actor association link	<actorLink>	*type* attribute to specify different types of actors' associations (e.g. is_part_of)

3.3.3 Specification of System Requirements using *i**

The *i** modelling language uses a collection of simple graphical objects to represent various components (*e.g.*, actors, goals, tasks, resources, etc.) of organizational environments and their information systems. Two types of diagrams – the strategic dependency (SD) diagrams and the strategic rationale (SR) diagrams – are used to depict the dependency relationships between actors in a system and the details of how other components are linked to one another. Although only a handful of component types are used for creating *i** diagrams, it is easy for new users to misinterpret how the modeling rules should be applied when moving between the SD and SR parts. We focused first on the SD diagram construction rules and quickly observed that although the individual rules are not complicated, when the diagrams to be reviewed or edited contain many components, it is easy for beginners to overlook errors that can cause problems in later development steps. Thus, we commenced by assisting new users with

gaining basic understanding of the SD diagram construction rules as it is described in the following sub-section.

3.3.3.1 Model quality checking for Strategic Dependency diagrams

By gathering information from the *i** Wiki portal [4], which provides a collection of documentation and modelling guidelines that are intended to help users create high quality models of system requirements, we composed as well as inferred a list of nine basic rule checks for *i** SD diagrams that students in an introductory training course would need to master at an early stage. Table 3.2 shows the results of testing a number of free *i** tools with features for checking model quality that could be used in such training.

Table 3.2 Basic SD rule checks implemented on freely available *i tools versus *i**Check**

Strategic Dependency (SD) rule checks	Tool A	Tool B	Tool C	*i**Check
1. Model containing only actors.	X	Δ	Δ	◯
2. Disconnected model elements.	◯	◯	X	◯
3. Dangling actors.	X	◯	◯	◯
4. Modeling colliding direction of dependency links (having same direction).	◯	X	Δ	◯
5. Modelling dependency link between actors without showing a dependum.	◯	◯	Δ	◯
6. Use other types of links (means-end, contribution, task decomposition) to denote dependency links.	◯	◯	◯	◯
7. Number of actors is less than 2.	◯	X	X	◯
8. Number of links for a dependency is less than 2.	◯	Δ	X	◯
9. Common dependum between multiple actors (connecting more than 2 actors).	X	◯	Δ	◯

In the table entries above, a circle ◯ indicates that the tool could either (primarily) prevent or detect the rule violation and provide a clear and lucid error message to indicate it to the user; a triangle Δ indicates that the violation (defect) is detected, but the tool does not provide a clear and straightforward message about what exactly the error is and how to fix it; and an X mark indicates that the violation is not addressed at all by the relevant tool.

Note that in a real industry setting, the errors listed above in the table 3.2 lower the quality of a model and may need several iterations of corrections before a useful representation of system requirements is finally generated for stakeholders discussions and reviews. So, these types of problems need to be identified and unlearned quickly. However, and as seen above, each tool is offering limited syntax checking features. Another point to mention is that the checks are scattered between the different tools (HiME, OpenOME and iStarTool) instead of being under the same platform. Therefore, our aim was to build and develop a piece of software that can integrate as many of these features as possible *i.e.* *i**Check.

To check his SD model, the novice learner will specify the SD checking feature and submit his request by clicking on the upload button to let the system proceed with first parsing the iStarML file and then performing the necessary checking. In case of diagramming defects, the checking functionality will return a list, a report-like feedback, involving the entire detected flaws details supplemented by correction-directed animations to facilitate the model rectification and improvement task.

3.3.3.2 Model quality checking for Strategic Rationale Diagrams

An ideal *i** requirements modeling tool should allow for complete diagrams' quality checking and assessment. In the previous sub-section, we reported on the first step towards developing our checker namely the Strategic Dependency (SD) model checking part specifically we mined from existing guidelines a list of checks concerning this type of *i** models. Hereafter, we focus on determining the different checks specific to the SR model *i.e.* the second type of *i** models checklist.

Dealing with the SR model necessitates more efforts as it is more refined and contains new additional as well as more complex concepts *i.e.* different refinement links. So, once again and from the *i** Wiki documentation, we derived and curated 13 construction rule checks concerning SR diagrams listed in Table 3.3 to be used in a basic first-level check of *i** models created by novice learners to the framework.

Table 3.3 below shows the results of testing a number of free *i** tools against a list of rule checks. As said before, the list of checks for SR (as was the case for SD) was mainly derived and concluded from the *i** wiki portal (*i** 1.0 guide) [4]. (The entries of table 3.3 here have same meaning as the ones in Table 3.2 that was dedicated for SD part).

Table 3.3 Basic SR model rule checks implemented on *i freely available tools versus *i**Check**

Strategic Rationale (SR) defect patterns (rule-violation checks)	Tool A	Tool B	Tool C	*i**Check
1) Dependency links used inside an Actor.	◯	Δ	Δ	◯
2) Including an Actor within another Actor.	◯	◯	Δ	◯
3) Strategic Dependency Link in an SR model is not connected to the correct "internal" element within the actor.	X	◯	X	◯
4) Disconnected elements within an Actor.	◯	X	X	◯
5) Drawing SR model internal elements outside the boundaries of the corresponding actors.	◯	X	X	◯
6) Extending decomposition Links beyond the boundaries of actors.	◯	X	X	◯
7) Decomposing goals by Task decomposition link.	◯	◯	◯	◯
8) Decomposing goals by contribution links.	◯	◯	◯	◯
9) Decomposing task by Means-End link.	X	X	X	◯
10) Decomposing task by contribution link.	◯	X	◯	◯
11) Decomposing Softgoal by Means-End link.	◯	X	X	◯
12) Decomposing Softgoal by task decomposition link.	◯	Δ	◯	◯
13) Softgoals and Goals are leaves (not refined further).	X	◯	X	◯

3.4 Evaluation of the *i**Check prototype

We used *i**Check to explore how different kinds of feedback can help the beginners recognize modeling mistakes at an early stage in *i** learning journey. By asking new learners to debug a series of *i** diagrams, we investigated the effectiveness of our tool in guiding them to locate and correct defects in given graphical schemas compared to some existing tools' checking features. We aimed to determine the factors that contribute to the success or failure of our solution. In the following subsection, we report on the very first experimental test that we performed to evaluate the early version of our tool which was mainly devoted for SD checking portion of the *i** model data.

3.4.1 Detection and correction of errors in *i** SD diagrams by novice learners using *i**Check

In developing the *i**Check system, we initially aimed to construct a thorough model quality checker of almost all specification concepts presented in the guidelines of the *i** Wiki guide for SD diagram construction. Preliminary tests of early versions of the tool indicated that although *i**Check returned a full set of model defects in textual format for beginners, sometimes an incomplete list of errors generated by some available tools was a more effective aid in completing the task because the messages were combined with visual markers of where the errors occurred. Since the *i**Check system is intended to work in complement with the available tools and not as a replacement for them, we expanded the development and followed up our tool to include GIF format animation for explaining how to remove and rectify specific errors in a model and this combination proved to be the most effective in reducing the confusion beginners experience in debugging exercises of defected requirements models.

In a series of tests, we measured how quickly and accurately engineering students with an introductory background on requirements engineering methodologies using the *i** framework are able to identify and correct modeling errors in a collection of SD diagrams such as the example in Figure 3.11 using a textual description of the system requirements and various types of syntax checking feedback to see how well each type is able to guide them through the tasks.

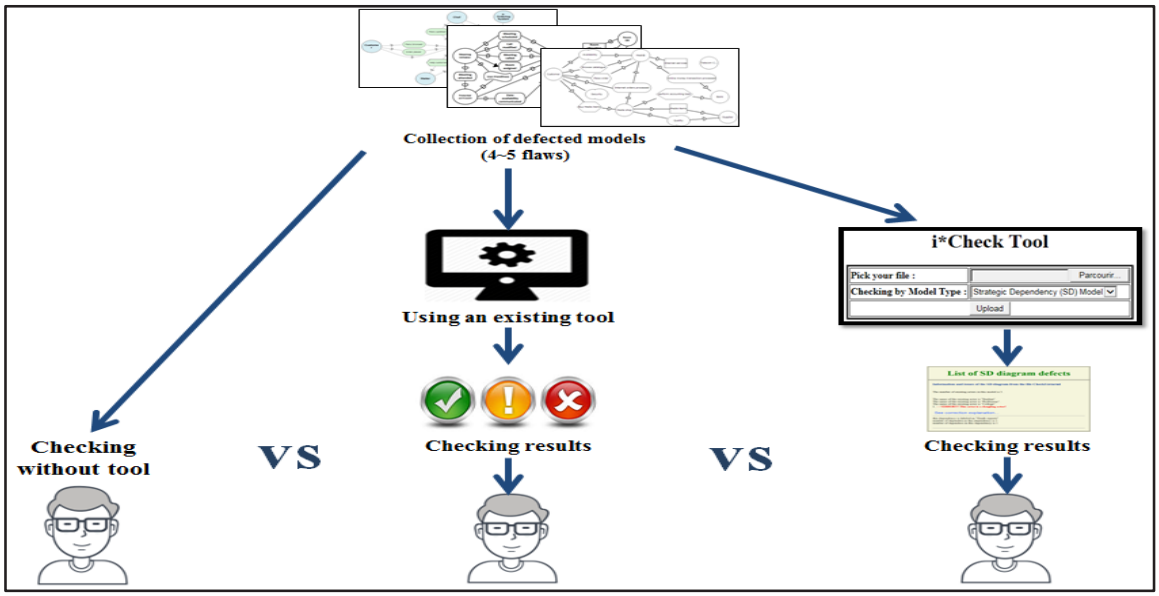


Figure 3.11: SD diagrams error debugging experiment with users receiving: 1) no feedback from existing tools, 2) available tool's feedback, and 3) *i*Check* rule violation feedback summary information.

Participants in this early investigational study and evaluation of *i**Check were undergraduate computer science and engineering students at Shinshu University (Nagano, Japan) with no previous experience with the *i** modelling language. Following an introductory tutorial on the *i** modelling language and framework, the participants were given a number of sample models to debug, using various types of syntax checking information from currently available systems and the *i**Check system that we developed. The test groups included: 1) NoTool users, who received no syntax checking feedback for the debugging task, 2) WithAvailTool users, who received feedback from an available tool that offers partial syntax checking features, and 3) WithI*Check users, those using the *i**Check support tool. In the test, the subjects of the NoTool group could rely only on their understanding and knowledge from the tutorial to do the debugging work. The WithAvailTool subjects were able to view carefully a snapshot of the syntax checking results of a selected tool as a reference in deciding how to correct the defects. The final group is the WithI*Check group that was able to use the online *i*Check* system to view the list of defects caught in the model and access the correction tips, which are in the form of GIF animations, as needed and as it is shown in the image in Figure 3.12 below.

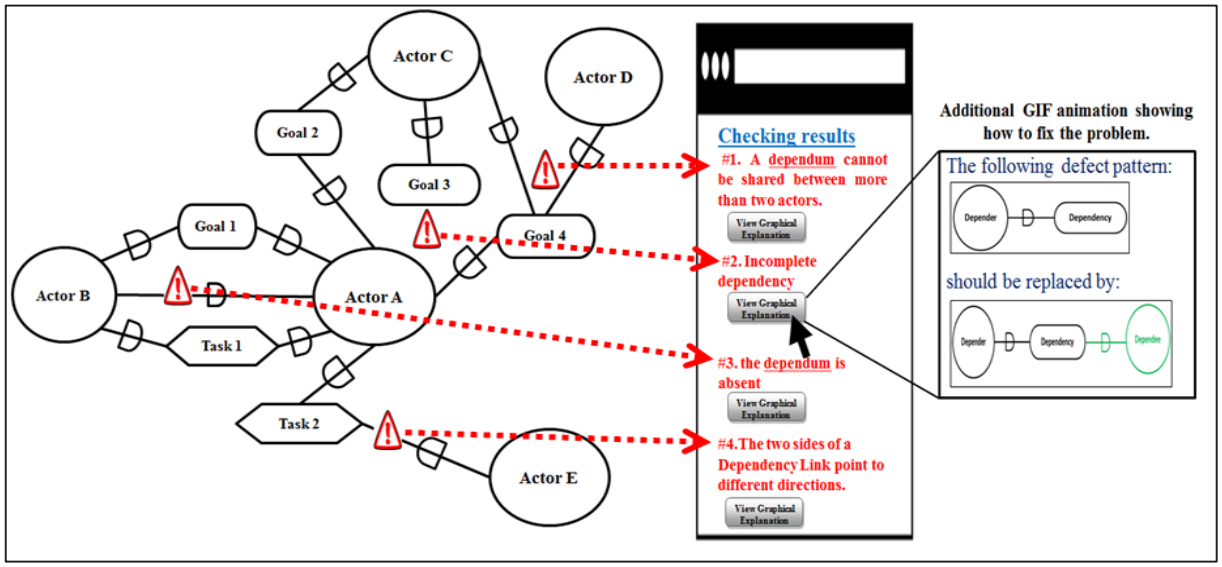


Figure 3.12: *i*Check* web tool checks for rule violations in an SD diagram and provides correction hints with text and GIF animation.

To begin, the subjects did a warmup exercise to be sure they understood the *i** model conctruction concepts to be judged and each subject had no difficulty in debugging the simple example. However, each model used in the actual test (e.g., Figure 3.12) contained 4 errors to be corrected and the average rates of success for each group are summarized in Table 3.4 along with the average time required to complete the debugging task.

Table 3.4 Results of *i SD diagram debugging tasks using various feedback sources**

Error Feedback Type	Number of participants	Average score (%) of debugging task	Average time for task completion
NoTool	4	45%	13.5 min.
WithAvailTool	4	50%	7 min.
WithI*Check	4	75%	13.5 min.

A post-test questionnaire was used to verify that subjects were fairly confident in their answers, but as seen in the results of the table, without a complete list of defects the novice learner is not able to sufficiently find and correct more than half of the detected errors even in a small-scale test model.

The average time for task completion with the *i**Check tool was the same as the case of having no error feedback references and we observed that the subjects spent most of this time viewing the looping and animated correction hints to achieve a high success rate in the specified task.

To sum up, the early system version's evaluation (2015) was promising and proved the potential that *i**Check bore to guide the novice learners throughout the checking and correction procedures.

3.4.2 Detection and correction of errors in *i** SR diagrams by novice learners using *i**Check

Encouraged by the first evaluation *of i**Check, we continued the development of the SR model checking part while adopting the usage of the same strategy of providing users with graphical GIF animations on how to resolve model defects and flaws. And subsequently, we used our prototype for checking the SR specifications of an *i** requirements model and testing its operation and usefulness as a tool that can be used in the classroom, the place where we expect tool assistants for syntax checking to be needed the most.

3.4.2.1 SR rule violation detection/correction tasks

During a short series of lectures to undergraduate and graduate computer science students on the *i** framework, including the construction rules for SD and SR diagrams, we conducted a series of tests in which the students were tasked with detecting and correcting errors in *i** model samples using three types of model checking: 1) no tool feedback ("WithoutTool"), 2) feedback from selected existing tools ("WithAvailTool"), and 3) feedback from selected existing tools together with *i**Check rule violation summaries ("WithAvailToolPlusI*Check") as shown in Figure 3.13 below.

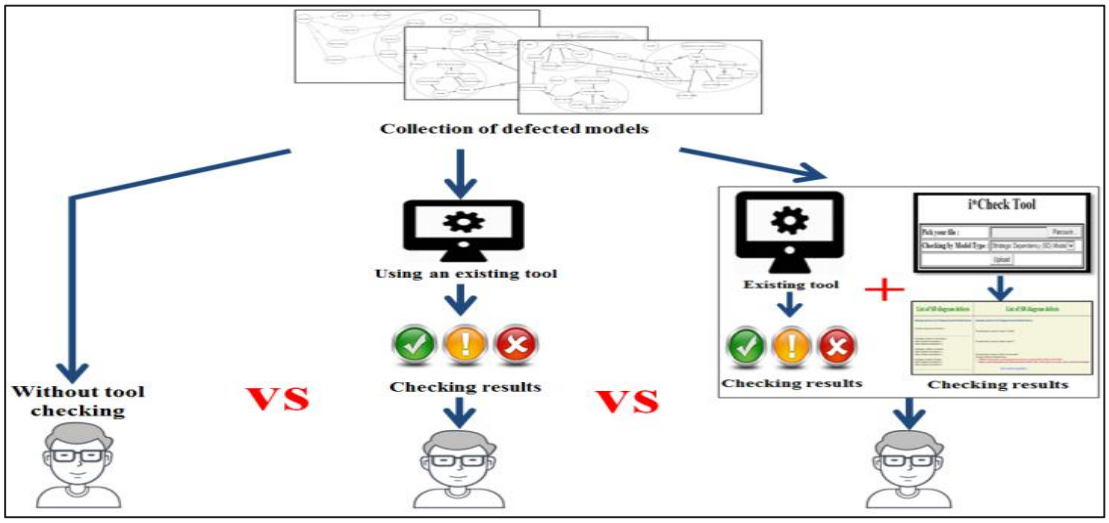


Figure 3.13: SR construct error debugging experiment with users receiving: 1) no feedback from existing tools, 2) some tool feedback, and 3) some tool feedback with *i*Check* rule violation summary information.

The experiments were conducted on three different days in lecture sessions of 2 hours each. For each experiment, we prepared three *i** diagram samples containing construction rules violations and asked the students to detect and correct as many as they could in a fixed time. For each test, we measured not only the accuracy of defect detection and correction, but also the time spent to accomplish each task.

The number of subjects for each experiment was three computer science students at Shinshu University just beginning training in the *i** framework. All experiments' subjects received the same tutorial to ensure that they acquired the necessary knowledge about the framework.

For each task, the subjects were asked to detect and correct as many construction rule violations as they could find in a printed *i** diagram containing 10 defects. The syntax checking feedback from an available tool was provided in a printed form. Besides, students were able to access the *i**Check feedback and correction suggestion animations directly through a PC web browser that displays it on demand. The given time for performing each task was 20 minutes.

3.4.2.2 Experimental Results

The test results for each day are shown in Tables 3.5-3.7 below. Each test was given on a different day, and with each passing trial, the time required for students to complete the debugging tasks to the best of their ability improved for all checking methods.

Table 3.5 Experiment Trial 1 Results

Checking method	WithoutTool	WithAvailTool	WithAvailTool PlusI*Check
Average number of Correct Answers	6.33	6	6.33
Average time (min.)	22	16	18.67

Table 3.6 Experiment Trial 2 Results

Checking method	WithoutTool	WithAvailTool	WithAvailTool PlusI*Check
Average number of Correct Answers	4.33	7	5.33
Average time (min.)	12.67	8.67	8.67

Table 3.7 Experiment Trial 3 Results

Checking method	WithoutTool	WithAvailTool	WithAvailTool PlusI*Check
Average number of Correct Answers	4.33	6	**7.33**
Average time (min.)	4.67	6.67	11.33

However, as the types of construction rule violations to be detected became more varied by the third trial, students who did the debugging work with no feedback from any tools (WithoutTool) could not find more than half of the basic errors and quickly realized (average of 4.67 mins. into the "given" 20 min. task) that they would not be able to do anymore corrections and gave up. The results for the WithAvailTool and WithAvailToolPlusI*Check students were nearly the same for the first trial, but the longer time spent can be attributed to the time required for playing the correction suggestions and hints animations on *i**Check during the debugging. By the third trial, as students became more accustomed to using the combination of feedback from the existing tools and *i**Check we find that they are able to correct 73% of the defects as compared to 43% and 60% for WithoutTool and WithAvailTool, respectively.

The experimental results also indicate that, day by day, there is a reduction in effort (time spent) to review and correct the defected models. Such decrease can be attributed to the *i**Check tool support usage as it helps students in steadily progress in grasping the *i** modeling language rules and good practices.

To illustrate, the benefits of our tool include a reduced effort to check and correct the defected models followed by an increased correct answers rate. Subsequently, the experimental results endorse the effectiveness and the impact of *i**Check usage on assimilating the framework concepts and its role to help the new users check their own models and improve their skills' level without asking for a coach's help.

3.5 Observations

As mentioned before in this thesis, some of the freely existing tools allow the exportation of an XML-based specification of the built *i** model aka iStarML. We did test our syntax checker prototype on a variety of iStarML format data files exported from these *i** authoring tools. Although some of the aforementioned tools included information specific to their operation (*e.g.*, placement of graphics, etc.), extracting the necessary SD related data for checking the SD part in particular was successful. While, in developing the SR checking part, we observed that some tools used different interpretations

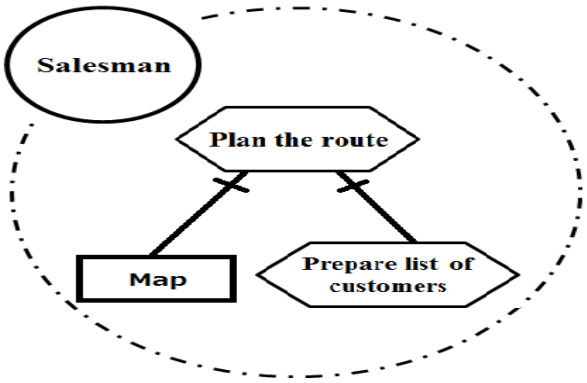
"representations"of the *i** constructs for SR diagrams which needed to be taken into a deep consideration for the prototype to work correctly.

3.5.1Discrepancies in interpretation of iStarML constructs

As described above, we used the iStarML description in [19] as the basis for developing the SR syntax checker proposed in this work. However, we discovered that among the tools that export their *i** model data in this format, some differences in the interpretation as well as the representation of the elements exist and we give an example in Figure 3.14.

Moreover, we tried to import an iStarML file which is generated by Tool A [5] and examine it using Tool B [7]. However, instead of displaying the expected model with its components and links, the output was a big mess with a lot of gaps and shortcomings which make it partial with a lot of missing graphical constructs. The tools are different in many structural and componential directions. Such format's difference hampers the files' interchange between both tools.

If we sketch the SR diagram components in Figure 3.14 (a) on different editors and export the data in iStarML format, we expect the same information to be contained in each file, but we discovered that this is not always the case. Figure 3.14 (b) and Figure 3.14 (c) show two different iStarML representations of the same diagram from two different tools. In the iStarML data of the first tool (HiME [5]), the Plan the Route task is decomposed into the Map resource and the Prepare list of customers subtask and the ielementLink descriptions are listed as children of the task being decomposed, namely the Plan the route element. However, in the second tool (OpenOME [7]), the ielementLinks are listed as children of the decomposing elements, *i.e.*, the Map resource and the Prepare list of customers subtask.



(a) Sample description of internal elements of an actor

```
<actor id="AA07" name="Salesman" type="generic" creation_date="1/5/16">
  <boundary>
    <ielement id="AA09" name="Plan the route" type="task"
creation_date="1/5/16">
      <ielementLink type="decomposition" iref="AA06" aref="AA07"
creation_date="1/5/16"/>
      <ielementLink type="decomposition" iref="AA10" aref="AA07"
creation_date="1/5/16"/>
    </ielement>
    <ielement id="AA06" name="Map" type="resource" creation_date="1/5/16"/>
    <ielement id="AA10" name="Prepare list of customers" type="task"
creation_date="1/5/16"/>
  </boundary>
</actor>
```

(b)

```
<actor id="_Kabk4GRVEeagAIS6fUotcA" name="Salesman">
  <boundary>
    <ielement type="task" id="_mn_U4GRXEeagAIS6fUotcA" name="plan the route">
    </ielement>
    <ielement type="resource" id="_r2n88GRXEeagAIS6fUotcA" name="Map">
      <ielementLink type="decomposition" value="and"
iref="_mn_U4GRXEeagAIS6fUotcA">
      </ielementLink>
    </ielement>
    <ielement type="task" id="_uhirEGRXEeagAIS6fUotcA" name="Prepare list of
customers">
      <ielementLink type="decomposition" value="and"
iref="_mn_U4GRXEeagAIS6fUotcA">
      </ielementLink>
    </ielement>
  </boundary>
</actor>
```

(c)

Figure 3.14: (a) Part of SR diagram showing the internal elements of an actor, specifically a task (Plan the route) decomposed to a subtask (Prepare list of customers) and a resource (Map). (b) iStarML output from Tool A (HiME) showing ielementLinks as children of the Plan the route task. (c) iStarML output of Tool B (OpenOME) where ielementLinks are given as children of decomposing ielements, *i.e.*, Map resource and Prepare list of customers task.

These types of differences not only create a problem in porting iStarML data between different authoring and editing tools, but also need to be checked carefully to be sure that our syntax checkers are extracting the information from iStarML files correctly. The current version of *i**Check assumes the Tool B's (OpenOME) iStarML style.

3.6 Summary

In this chapter, we started by giving an overview of the freely available *i** modeling tools and showed evidences of their limitations concerning the syntactical checking functionalities. Thus, we highlighted and justified the need for a "complementary" tool to the already existent tools which unfortunately do not offer the very complete and comprehensive set of checking rules and sometimes even if they do, the returned error message is poorly constructed and explained and this may lead to novice learner's confusion and misunderstanding. Then, we presented in details the web-based *i*Check* system which is developed to help *i** users detect and correct syntax errors in the strategic dependency (SD) specifications of their requirements models as well as in the detailed (SR) portion of *i** diagrams. From the *i** Wiki documentation, we compiled a list of 9 basic (SD) and 13 basic (SR) construction and syntax rule checks that should be done in every *i** model created by novice learners. Since we tested a number of freely available *i** authoring tools, we found out that none of them provided syntax checking features for the full set of rules and conventions. Rather than create a completely new authoring tool, we designed *i**Check to work as a tool assistant to the existing systems. By using the iStarML (XML) data files exported by the available modeling tools as input and producing a summary of the (SD) and (SR) construction rule violations detected along with suggestions and hints in animation form on how to rectify the problems. Eventually, we consider our tool as an explanatory tool that provides students with further information about their models, the rules that were violated and by then the correction steps by means of GIF animations which are presented to the system user to guide him through his model review and rectification.

One important discovery during the development of *i**Check was that different *i** tools sometimes used different interpretations of how to document the elements particularly those of SR specifications in an *i** model in iStarML format. That is, the same *i** model created on different tools could result in different exported iStarML files' styles. This problem needs to be addressed further not only to solve portability problems, but also because it affects how *i**Check extracts SR data from the exported files.

Finally, the testing and evaluation of the *i*Check* syntax checking functions with novice learners to the framework proved the effectiveness of our proposed tool.

Chapter 4. Supporting the iStar Model Content Review and Validation

In the previous chapter, we have seen the proposed solution for the *i** (iStar) model quality undetected or poorly expressed feedback for syntactical defects. In this chapter, we present our approach to address the second important research thread *i.e.* the lack of *i** model content review for the subsequent validation.

4.1 The lack of *i** model review for validation

When teaching a new paradigm which involves a practical training to a large group of students, it often becomes time-consuming and impractical for a single instructor to give advice on an individual basis on how to correct errors being made or to evaluate the model informational content for validation purpose and subsequently, the need for computer-aided assistants arises.

In Chapter 3, we discussed the necessity of having a tool that covers almost the complete set of *i** modeling rule checks and we reported on our proposed tool's, *i**Check, development work and its evaluation results. Although *i**Check tool was useful in showing novice learners how to edit their models to make them free of syntax errors, there were a number of situations in which they could not recognize the semantical flaws and defects (the second part of model quality issues) of a model using the feedback from our *i**Check tool. Put differently, in continued observations of novice learners constructing and checking *i** models, we discovered that although the *i**Check tool feedback was effective in guiding beginners on how to correct basic syntax errors, several situations occurred in which users missed errors concerned with the informational contents of diagrams, *i.e.*, the novices grasped the mechanics for sketching diagrams, but did not realize that they had misrepresented or added unnecessary information to the built requirements model.

After practicing how to gather requirements from a natural language description in documentations or from interviews/workshops and constructing an *i** model of the collected information, the beginners needed a simple way to confirm that the information contained in the created models matched their understanding of the requirements, with no omissions or unnecessary additions. Put differently, capturing knowledge in *i** goal models is a critical task that must be discussed, understood, reviewed, evaluated, questioned and validated by all the different stakeholders ranging from domain experts to technical people aka requirements engineers (IT experts) to ensure that the embodied information in the model is adequate. Misunderstandings lead to approval of immature or incorrect models. And such produced models can lead to additional effort to perform in later stages when they face the reality.

4.1.1 Examples of observed model content deviations

In the following sub-section, we present two examples that show how an automatic summary of the contents of an *i** diagram organized in an outline form would be useful for beginners who need to revisit the original descriptions of the software requirements of a system and validate that their constructed models are error-free in a semantical sense.

4.1.1.1 Misrepresentation of intended requirements

Although *i** uses only a small set of graphical constructs to represent the actors involved in a system, their dependency relationships, and the breakdown of their goal oriented requirements, some practical training time is necessary for beginners to grasp how to use the various constructs and rules to represent

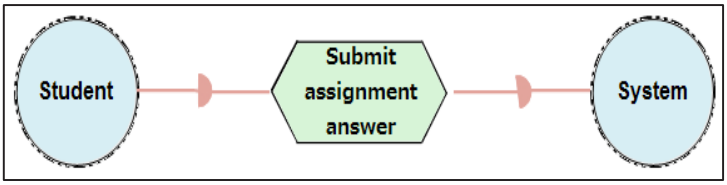


Figure 4.1: Example of wrong direction of the dependency relationship. There is no syntactical error in the application of the design rule, but the intended meaning ("The system depends on the student to submit an assignment answer.") is depicted in the reverse direction.

this information. In the SD portion for an *i** model, the main job is to identify all of the actors in a system and document what each will need provided or fulfilled by others such as in Figure 4.1 above. For the beginner, the SD construction rules are not difficult to apply, but in the course of model construction, humans sometimes introduce errors (either out of misunderstandings or out of carelessness).

In Figure 4.1, for example, a requirement stating, "The system depends on the student to submit an assignment answer" is written in the reverse direction. The format of the dependency specification is not in error so there will be no warning by *i** syntax checkers, but the meaning is incorrect. Although this type of error could be caught with careful review, as a model grows in size and becomes complex, it will be easier for a human to mistakenly overlook such an error and propagate it to later stages of development (overloaded diagrams complicate the perception of information). This can cause misunderstanding and confusion issues and costly corrections that the requirements engineers are particularly employed to avoid.

If this misrepresentation is not corrected at the SD diagram level, the next step of specifying and refining the goals of each actor, *i.e.*, the SR diagram construction, will proceed and it will become more difficult to spot the error. If the SD level misrepresentation of information shown in Figure 4.1 is left uncorrected and the modelling process continues to formulate the expanded SR model, the chances of a human catching the error from doing just a manual check of the information will not be high. Even an examination of the human-readable istarML (XML-format) data of the model (Figure 4.2) will not easily detect the problem since a human must often sift through and jump to different locations of the file just to recognize one piece of information, in this case that a student and the system to be developed have a dependency relationship in which the system depends on the student to take action on submitting assignment answers.

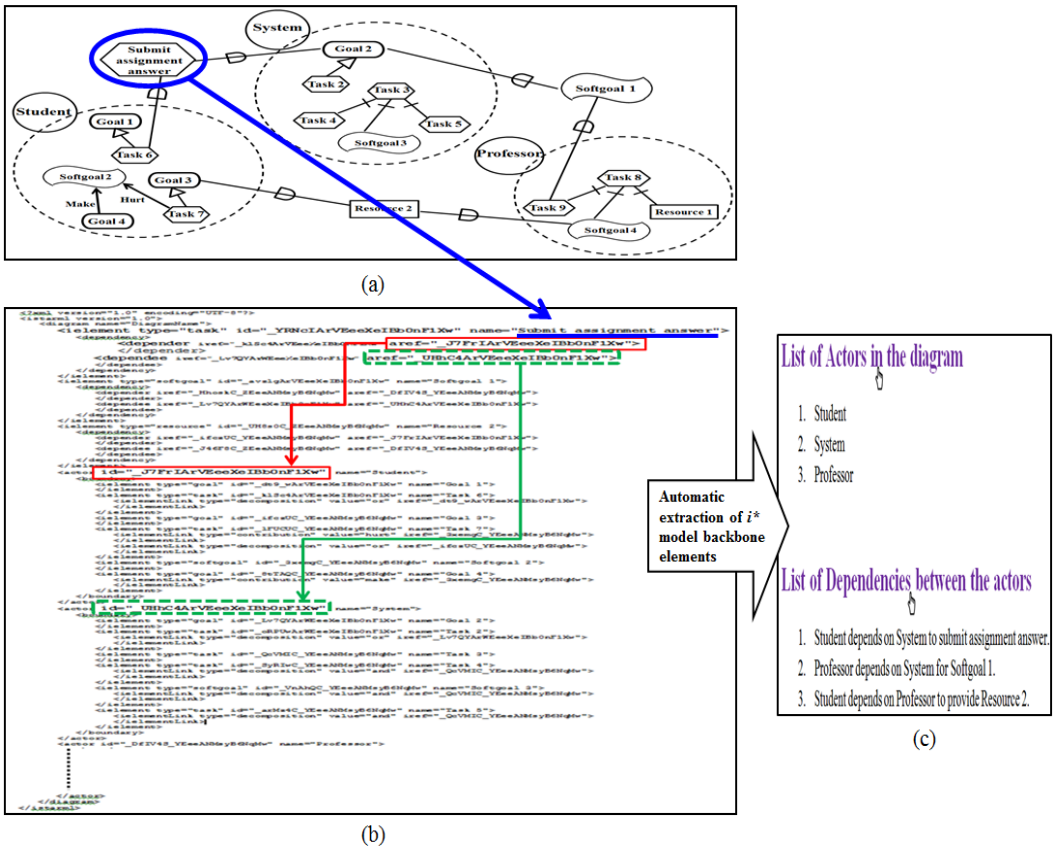


Figure 4.2: Checking the meaning of each piece of information in a requirements model can become tedious for a human when (a) the number of components in a model grows and (b) tracing the human-readable iStarML data of the model requires jumping to different areas of the file to compile one piece of information. (c) An automatic summary generator extracting the i* model backbone elements from the XML data and listing the basic requirements information in short natural language sentences could help novices more easily find errors in the model contents.

In our observations of *i** model construction and checking works by novice users, it became apparent that a simple summarization of the backbone elements of an *i** model could be helpful in reducing the time and effort of checking the meaning of the information expressed. So, we proposed an automatic tool like *i**Check which can work online and take as input iStarML data of models generated from various modeling tools and editors to produce short natural language descriptions of the *i** model backbone elements in a simple table of contents (TOC) style (Figure 4.2(c)). In this way, beginners to the modeling domain can have a more human-friendly document to check for discrepancies in the model description.

4.1.1.2 Misinterpretation of design constructs

The second example of novice users having problems recognizing how to specify requirements in *i** occurred in the goal refinement work of SR diagram construction. In this situation, users learned that the Means-End link is the modeling construct for presenting the alternative ways of achieving a goal. The examples in the tutorial lesson always showed multiple alternatives and the guidelines did not explicitly define the cardinality (1..n) of the construct so it was not completely clear to some students that it is

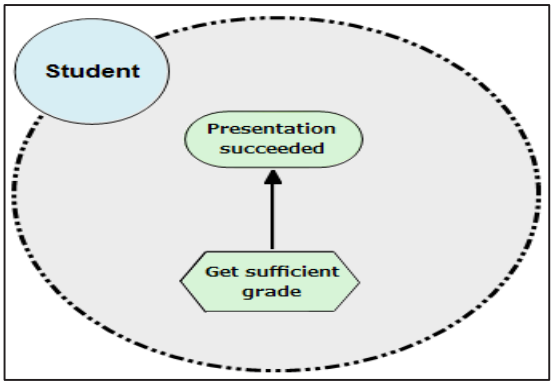


Figure 4.3: Example of goal refinement showing that a single means is the only (sufficient) way to achieve the goal. Beginners forcing multiple alternatives to be listed for each goal will see no syntax errors in their diagrams, but may be introducing irrelevant components without realizing it.

possible to have only one means for achieving a goal as shown in the example of Figure 4.3.

This resulted in some students forcing model edits to create a situation in which each goal had at least two alternatives listed, even if such information was not contained in any of the requirements of the system. As with the example in the previous sub-section, a syntax check of the model will declare that it is free of design errors, but in fact irrelevant components exist and some human judgment will be necessary to determine whether or not the information contained in a model is enough or too much. Such misinterpretations of the model construction rules in a new modeling paradigm occur frequently with new users. So, again, the support beyond simple syntax checking is necessary to detect this type of error. The automatic generation of textual summaries of *i** models can be a useful tool for other persons reviewing the models constructed by beginners to check their meanings.

4.2 GENERAT*i**ON: a tool to assist beginners in reviewing and validating *iStar* diagrams' contents

To address the aforementioned problem (the second problem of this thesis), we proposed the development of a tool assistant, called GENERAT*i**ON [35], which can be used in combination with our *i*Check* tool which also accepts an iStarML (XML representation of an *i** diagram) file as input to generate a textual summary of the backbone elements of a model in short natural language sentences organized in a table of contents style.

Despite there seems to be a little work in the literature, we found a work on the development of an Eclipse based tool [37] which supports a proposed programming-like textual syntax language for Goal-oriented Requirement Language (GRL), an *i** variant, which allows for both the creation and modification of large models. However, a simple translation into natural language, rather than programming language, will suffice for enhancing the error detection ability of novices in the scenarios presented in the examples of this thesis.

The most important elements of the *i** SD diagram include a list of the actors in a system and the dependencies between them, *i.e.*, what each actor needs provided or accomplished by the other actors, expressed in short statements of the form "A depends on B for…". Such a summary can be a useful, complementary document to the *i** diagrams for beginners to quickly review their models for semantical errors such as unintentional reversals of meaning, omissions of information, inclusions of irrelevant information, etc. As the SR diagram components add complexity to an *i** model, the automatically generated textual annotations of the contents by the tool assistant proposed in this work will be even more useful. The summaries can be used not only by novice learners to check the meanings of model constructs, but also by other persons such as non-technical stakeholders who are involved in discussions and decisions concerning the requirements of a "future" system and need to review the model contents although they are not specialists in the modeling paradigm.

We advocate the importance as well as the efficiency of our tool as it is convenient and appropriate to translate an *i** model, its structure (The parent/child relationship resulted for any refinement type) and the wording used in labeling its involved conceptual constructs to a more human readable *i.e.* more "natural language" format. As models may grow complex in a very rapid way and with the absence of *i** modularity, we believe in the added value that novices can get when they have both views of the very same system story (domain under study) so they can compare the two versions (representations) namely

the graphical and textual (natural language NL descriptions) to ascertain that what they really meant in the first place is present in the model and its accompanying and corresponding table of contents (TOC).

Our aim is to assist the novice learners of the *i** language to carefully review and check the meaning and the content details of their created models in order to increase students' comprehension on software requirements models particularly the *i** models by deriving tables of contents reflecting the informational and structural details that reside in the models while in a very organized and indented way. Thus, we let them (the beginners) introduce corrections and improvements in case of detected semantic mistakes. Put differently, we aspire to allow the students to get a structured view where they can access and navigate to see the different "model" elements grouped by type (from high level: actors, dependency to more detailed level: refined internal elements and links connecting them) in a very elegant way. In other words, we assume that such grouping will allow the beginner to easily figure out and detect any misinterpretation, misunderstanding or misrepresentation of the original set of requirements in the system story.

4.2.1 Development description

The GENERAT*i**ON tool assistant was developed with web-based technologies, the PHP language, in order to take advantage of recent advances on web browser performance and the provided accessibility and most importantly, it benefited again from the offered parsing functionalities of DOM XML.

Initially, the models of our concern are represented in their native format since they are designed using the existing diagramming editors. These models are either translated into XML format or they are manually created from scratch by the student (which risks being an error-prone and tedious work). Then, these iStarML files are used as inputs to the table of contents generator aka GENERAT*i**ON as it was the case for *i**Check system. Once uploaded, the iStarML version of the model will be analyzed and parsed and then a hyperlinked version (of the model) will be returned to the user. This result will report information on every actor, dependency, intentional element and links existing in the model. Figure 4.4 illustrates an overview of the role played by our proposed tool.

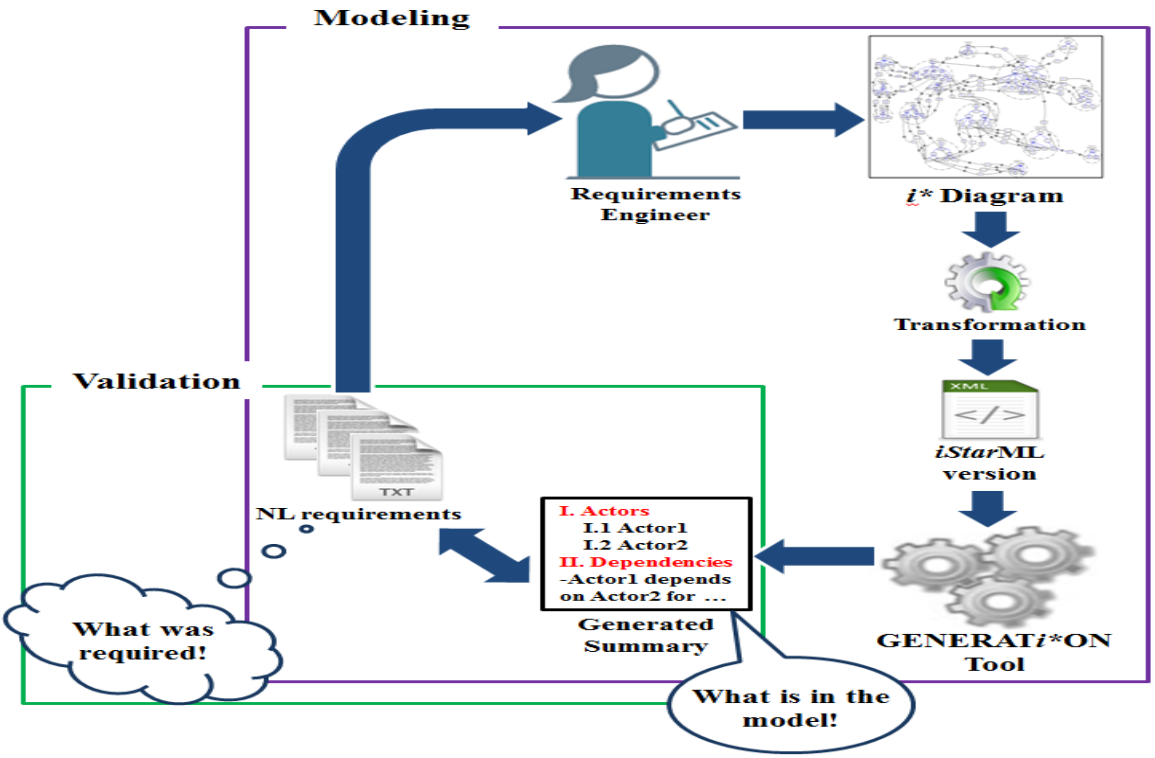


Figure 4.4: An overview of the model validation activity using the GENERAT*iON tool**

By developing our tool, we do not intend to deny the expressivity and the intuition that do offer the visual representations of *i** requirements, in the contrary, we consider the derived table of contents as a mirror that reflects and reproduces what is being expressed inside the diagram. Moreover, we resorted to table of contents generation approach in order to alleviate to some extent the difficulty of *i** model navigation and pursue for example, in case of large and evolving SR model with insufficient space (unclear structure caused by intersected links etc…), elements refining some higher level goal or task.

We seek a lot of advantages out of our approach, as our online tool automatically creates and gives back the table of contents any time on demand to help the novices understand the technique especially when the models get complex or more generally to any non-technical *i.e.* non-expert person who doesn't have the required skills to understand *i** goal models (may prefer the textual representation) and so he will find our approach easier as it gives him the chance to inspect and detect any misinterpretation, misrepresentations (of the domain under study) to later introduce modifications, improvements or corrections to his design if necessary (localizing problematic parts of the models helps stakeholders focus their discussion and take corrective actions). In other words, it is of great importance that the various stakeholders from different backgrounds carefully review, evaluate, correct, and propose improvements to these sketched *i** models.

For example, a complex *i** model in its diagrammatic representation is not easily understood and the navigation through it is time consuming whereas, inspecting the generated TOC could be useful and helpful in pinpointing omissions, misrepresentations or unnecessary additions to the requirements model. Thus, the ability to automatically derive a TOC makes our system a useful tool for debugging any *i** model.

The generated table of contents (TOC) by our prototype tool is delivered as precise, verbose and complete as the original *i** model from which it is derived and subsequently, each time the *i** goal model is changed, saved and exported to iStarML (the same process), the new (modified) TOC can be re-derived once again in order to keep in synchronization with the evolving model. Furthermore, it lists the names of the actors, intentional elements (either internal elements or dependums) from the *i** requirements model and it utilizes the same icons used in the diagrammatic representation for each concept. Such icons can be helpful for users to remind them of the concepts' types. They also provide some visual grouping of the model constructs by their types. The indentation in the table of contents appears by level, after clicking on a certain hyperlink.

4.2.1.1 Suggested and developed generation templates

*i** goal models are often used to support the understanding, the modelling and the analysis of organizational environments and their information systems. The accuracy of such models usually requires that domain experts (staff of the organization) carefully review, revise, evaluate, correct, and why not propose improvements to these models. However, domain experts are often not experts in (conceptual) modeling. Consequently, they may not have the skills to understand, for example, in our case the *i** models except at a relatively superficial level.

Generally, we believe that the validation of requirements models using one readable representation *i.e.* textual description, is useful because all stakeholders (novice learners like students, business people and non-technical staff, whom are inexperienced users of the *i** methodology and modeling language and its tooling) will be more comfortable and confident to understand the discourse in natural language.

To this respect, we define an approach for generating (natural language) textual descriptions based on *i** goal models. The approach consists in a set of phrasing templates (patterns implemented in PHP codes) dedicated to each type of *i** models, namely the SD and SR models.

- ✧ **In case of the Strategic Dependency (SD) model:**

Given that this model focuses on the external relationships (dependencies) among actors that provide a more abstract view of the reality. There exist 4 types of dependencies: goal, softgoal, task and resource.

The goal here is to translate each relationship (dependency) type between two actors into an equivalent and corresponding pattern that verbalizes and expresses it firmly (nothing less nothing more).

The candidate 'Task dependency' phrasing template:

-<Actor1: the depender> depends on <Actor2: the dependee> to <Task dependency (label)>

Example: Meeting initiator depends on Meeting participant to send the preferred dates.

The candidate 'Resource dependency' phrasing template:

-<Actor1: the depender > shall receive <Resource dependency (label)> from <Actor2: the dependee >

Example: Meeting Participant shall receive proposed date from Meeting Initiator.

The candidate 'Goal dependency' phrasing template:

-<Actor1: the depender> depends on <Actor2: the dependee> to achieve the goal <Goal dependency (label)>

Example: Patient depends on Doctor to achieve the goal medical care received.

The candidate 'Softgoal dependency' phrasing templates:

-<Actor1: the depender> depends on <Actor2: the depend> to satisfy the softgoal <Sofgoal dependency (label) >

Example: Patient relies on Medical Professional to satisfy the softgoal privacy of personal data.

In case of the Strategic Rationale (SR) model:

The SR model provides a more detailed level of modeling by looking "inside" the actors to model internal intentional elements and the different relationships (refinements) among them.

Therefore, the textual generation task here is performed per actor.

The starting point will be the generation of all different internal elements that reside inside the actor boundary using the following template:

The internal elements of the actor <The actor Name> are grouped by type:

-<List of goals (labels) >

-<List of softgoals (labels) >

-<List of tasks (labels) >

-<List of resources (labels) >

Phrasing templates for each Intentional Elements Relationship (different refinements types) concerning only the internal elements which are generated in a hyperlinked format.

- In case of Means-End relationships:
  - To achieve the goal <Goal label>, one of the following elements (alternatives) must be accomplished:
    -<[List of] Tasks labels>

  Example: To achieve the goal "Air ticket purchased", one of the following elements (alternatives) must be accomplished:
  -Buy the ticket online
  -Buy the ticket via a travel agency

- In case of Task-Decomposition links (AND Decomposition):
  - To perform <Task label>, all the following elements must exist:
    -<[List of] different internal elements (subgoals, softgoals, subtasks, resources) labels >

    Example: To perform "Provide book information", all the following elements must exist:

    -Extract book data
    -Provide info by title
    -Book

➢ In case of Contribution links:

The following elements (which can be goal, softgoal, task and resource) <Intentional Element label > contribute to:

- <Softgoal label> Value (Make, Help, Some +, Break, Hurt, Some-, Unkown, AND, OR)

Example: The following elements contribute to "Attract more customers":

-Sell the books online help

-Offer some coupons help

Evidently, obtaining a high quality *i** goal models minimizes flaws percentage in later software development phases and helps the modeler to produce an *i** model that is easier to understand and maintain.

4.3 Summary

The aim of this chapter is to describe our developed table of contents (TOC) generation tool and also to show the benefits of having a graphical version depicting the system story under study consolidated with a more human oriented and well-structured version *i.e.* maintaining both graphical representation alongside with the textual format to finally and accurately review and validate the desired model. Put differently, the generated TOC of an *i** model is beneficial to different stakeholders. However, it is certainly not a replacement for the diagrammatic view as we believe that the two representations complement each other.

We assume that our prototype provides an added value in terms of understanding the *i** models as it is designated to complement the visual representations of the *i** models and thus allow their accessibility to various stakeholders with different background and training. The current version of the GENERATi*ON application assumes and works best for the H*i*ME style of the input iStarML. We seek one more advantage from our prototype. Since, for example, it returns the intentional elements grouped by their types (by goal, by task, etc.), this will facilitate the checking of the conventional labeling styles provided for each intentional element type (*e.g.* a task should be specified and written as follows: task → verb + object).

One obvious disadvantage (weak point) of the generated TOC may be its increasing size (after each click, new information will be displayed on PC screen). However, this is inevitable since all generated information results from the elements, their textual annotations and the linking which binds between them *i.e.* captured from the created model.

Chapter 5. Conclusion and Future Work

This chapter summarizes the entire thesis and sheds light on possible future research topics that can be performed.

5.1 Synthesis

In general, modelling has been advocated as a very important part of software development, particularly in the requirements engineering (RE) phase, in order to tackle complexity by providing abstractions and hiding technical details. Thus, due to the wide and increasing application of modelling in the past few decades, numerous formal and semi-formal approaches to modelling have been developed such as the *i** (iStar) framework.

Now, *i** is one of the most widespread, best positioned and well-grounded goal modeling languages and methodologies. Like any modeling language and technique, *i** has a large documentation covering the basic ingredients for modeling rules, guidelines, conventions and best practices on how to select the language constructs with an overall lower complexity as well as define the patterns for better usage and combination of those constructs and concepts. Obviously, *i** has been supported by a vast range of tools available, enabling its widespread adoption and practice usage, to permit the practitioners (experts, instructors and students) sketch their requirements models. Despite the aforementioned positive points, it is not straightforward that examining a graphical description could allow users, in particular novice learners, to easily discover modeling errors (*e.g.* disconnected elements, absent dependum, *etc*.).

Choosing a certain tool to use when teaching the *i** language is challenging and delicate, as instructor has to be careful about his choice since surveying the state-of-the-art tools revealed some weaknesses, despite their merits, that would affect and impact upon the quality of produced models. In other words, we performed an analysis of a set of *i** Wiki-derived checks for a specific set of freely available tools and we

could find out their strong points as well as their shortcomings and imperfections. The aforementioned defects are represented in two main categories: the ones related to SD model and the ones that specifically concerned the SR model.

The evidence that we got from the performed tools' survey settled down the framework of this research work through the definition of the first research problem and one of the major contributions expected from this dissertation.

First aim of our research work is to face model syntactical quality issues (not respecting nor adhering to the *i** modeling language rules, guidelines and best practices) on *i** models raised by the usage of the existent systems and their checking features. Those issues can be tackled facing the before mentioned sub-problem *i.e.* limitation of checking features provided by the freely available tools.

Our second goal in this thesis work is to allow the model creators review their built models for subsequent validation of their contents correctness and compliance with domain knowledge. Such validation is so important since it enables the modelers communicate what has been expressed in their models. To put it in another way, we desire to also detect and clear errors other than syntactical ones (those which cannot be fixed in the same way and strategy like the syntax defects require).

So, the main contribution intended by this dissertation was the improving of the quality of iStar models at the analysis phase by developing a web-based toolkit that supports *i** modelers in reviewing their models syntactical quality at a first step and then helps them in validating what has been described in their graphical depictions. It is important to mention here that our intention was not to suggest how the RE, in particular the *i** methodology subject, should be introduced and taught effectively in a classroom setting since we know that the learning process is based on many phases. Instead, our goal is to assist the novices in reviewing their built models to make sure of both their syntactical quality and valid correct contents without the help of a nearby coach or teaching assistant.

5.2 Contributions

In the following sub-sections we summarize the major contributions of our dissertation.

5.2.1 *i**Check: an online free tool to detect *i** models defects

Aware of the mentioned limitation revealed in the tools' assessment *i.e.* limited model checking features (since the principal consideration that influenced the development of our tool was that the existent tools' error messages could, and should be improved upon) which is also scattered between the three

different surveyed systems, we developed an online system which is envisioned to circumvent this gap and complement the existing *i** tool landscape as it will only offer checking functionality to the input *i** diagrams' iStarML representation. To say it in simple way, we provided and developed an online "separate and independent" model quality checker designated to produce a report like feedback that describes the detected (if any) defects and gives hints and suggestions on how to correct them. Thus, we aim to support the *i** language learners and users to produce good quality models as, in case of detected model flaws and rule violations, the modelers will iteratively introduce the necessary corrections and improvements until they get models totally free of any error and rule violations. Our work can be considered as a resource for instructors of introductory RE courses, using the *i** approach, and why not developers of tools which are designated for novice modelers.

*i** tools generally allow for modeling artifacts and some of them enable the interchange and interoperability among different tools. Most of the interchanged models use a serial representation of *i** requirement model through a dialect of XML called iStarML. We make use of this interchange format as an input to our prototype aka *i**Check which will be parsed in order to fetch for any possible modeling faults (*i.e. i** model checking must be preceded by either a model exportation from a tool offering the functionality that is allowing and supporting the interoperability format aka iStarML or a manual iStarML file construction by the novice modeler).

The returned checking results consist in two types of feedback: one is textual error description (error display) and GIF animations (fix suggestions). We assume that the latter one is just needed in the very beginning of the learning journey and at some point the novice learner will be accustomed and familiar with the necessary correction steps and thus, he will be satisfied only with the textual error messages as we suppose they will be enough for the learner to introduce the required fixes.

Note that the detected defects along with their corresponding exigent corrections can change a model that is not fully well-structured into a well-structured and sound one without changing its (yet informal) semantics and thus they help novices avoid any possible source of misunderstandings and wrongful implementations later.

The effectiveness of our proposed tool was empirically tested by means of experimentations which returned positive and promising results of using *i**Check as a complementary tool to the available systems.

5.2.2 GENERAT*i**ON: an approach for model content review and validation

Besides model syntactical quality, the produced *i** models would also reveal other types of shortcomings and imperfections, due to misinterpretation, misunderstanding (switching roles, omitting something because of carelessness) and even because of total freedom when expressing the under study domain knowledge.

For this purpose and as was previously mentioned, we developed an online application called GENERAT*i**ON which also (same strategy that is used by *i**Check) accepts an iStarML version of the created *i** model, parses it to extract specific data and then generates a table of contents that describes and reflects the content of the model in a structured way. Put differently, it allows expressing and translating the different types of elements of the model including constructs, links and relationships connecting them, which are visually expressed along with their textual annotations, to a human understandable manner.

5.3 Future work

Inspired by the proposal in this thesis, we have identified some additional research directions that could be extensions to the actual work or new perspectives for new research lines. In other words, there are several interesting avenues for future work.

The first research direction could be continuing the improvement of our current toolkit by addressing the limitation of our current tool *i.e.* not covering all the *i** framework constructs.

The second research direction, for improving the current work, includes the extension of our prototypes in the following ways: update and upgrade our *i**Check tool and allow it to offer the checking of best-practices rules reinforcement of iStar 2.0 (released in mid-2016). Following the same strategy, this tool should provide hints (a depiction of a model with the error found and its corrected version) and guide the modelers in the overcoming of non-compliances in *i** models. Obviously, GENERAT*i**ON tool support will also be a subject of such updating.

Although we have performed case studies to evaluate each part of the thesis, we have acknowledged the needs of collecting further empirical evidence in regard to the follow aspects: firstly, the case studies we have performed within laboratory experimentations which provide preliminary evidence that our approaches can be adopted in reality *i.e.* in real classroom setting. However, we need to further evaluate our approach with a significant number of participants. To this end, controlled experiments would be the ideal empirical method to collect such empirical evidence. Another idea is to separate the effectiveness' evaluation of the *i**Check system by investigating in an independent manner the efficiency of each feedback type, *i.e.* the returned textual descriptions of detected errors versus the GIF animations errors' fixing hints, and test how, how fast and which one of them is more helpful and useful than the other.

Additionally, extending the model checking facility (especially for *i**Check) in order to register the history of modeling errors introduced by the user and providing the functionality to the user to allow him generate reports with this information and understand in which *i** model scopes he introduces more errors and exactly in which type of *i** elements, so the user can improve his knowledge in this area, enhance his *i** modeling skills and potentially avoid introducing these errors in future, can be another perspective for further research work.

References

[1] E.S.K. Yu. " Towards modelling and reasoning support for early-phase requirements engineering", in Proceedings of the Third IEEE International Symposium on Requirements Engineering, pages 226–235. IEEE, 1997.

[2] E. Yu. "Modeling Strategic Relationships for Process Reengineering". PhD thesis, Department of Computer Science, University of Toronto, Canada, 1995.

[3] J. Horkoff. "Observational Studies of new i* Users: Challenges and Recommendations". in *Proc. iStarT*, 2015, pp. 13-18.

[4] i* Wiki Guide, http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Guide .

[5] HiME: Hierarchical i-star Modelling Editor, http://www.upc.edu/gessi/istar/tools/hime/index.html .

[6] Á. Malta, M. Soares, E. Santos , J. Paes , F. Alencar and J. Castro. "iStarTool: Modeling requirements using the i* Framework". *CEUR Proceedings of the 5th International i* Workshop* (iStar 2011), Trento, Italy, pp. 163-165.

[7] OpenOME, an open-source requirements engineering tool, http://www.cs.toronto.edu/km/openome/ .

[8] R. Laue, A. Storch. "A Flexible Approach for Validating i* Models". *CEUR Proceedings of the 5th International i* Workshop* (iStar 2011), Trento, Italy, pp. 32-36.

[9] Alencar F., Castro J., "Integrating Early and Late-phase Requirements: A Factory Case Study".

[10] Yu E., "Towards Modelling and Reasoning support for Early-Phase Requirements Engineering". Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97) pp. 226-235 .IEEE Computer Society Washington, DC, USA 97.

[11] Lamsweerde A., "Goal-Oriented Requirements Engineering: A Guided Tour". Invited minitutorial, Proceeding 5th IEEE International Symposium on (RE'01), Toronto, IEEE, August 2001, pp. 249-263.

[12] Letier E., Lamsweerde A., "Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering". Proceedings of FSE'04, 12th ACM International Symp. On the Foundations of Software Engineering, Newport Beach (CA), Nov. 2004, pp. 53-62.

[13] Dardenne A., Lamsweerde A., and Fickas S., "Goal Directed Requirements Acquisition". Science of

Computer Programming, vol. April 1993, Pp. 3-50.

[14] Potts C., Takahashi K. and Anton A. I., "Inquiry-Based Requirements Analysis". IEEE Software, March 1994. pp.21-32.

[15] Anton. I. A. and Potts C. "The Use of Goals to Surface Requirements for Evolving Systems". *International Conference on Software Engineering (ICSE `98),* Kyoto, Japan, pp. 157-166, 19-25 April 1998.

[16] Mylopoulos J., Chung L., Liao S.S.Y., Wang H. and Yu E., (1992, June). "Exploring Alternatives During Requirements Analysis". IEEE Software, Volume (18), Issue: 1, Jan/Feb 2001, pp. 92 - 96.

[17] Cares C., Franch X., Lopez L., Marco J., "Definition and Uses of the *i** Metamodel".

[18] Horkoff J. M., "Using i* Models for Evaluation". Master Thesis (2006), Department of Computer Sciences, University of Toronto.

[19] iStarML: The i* Mark-up Language guide, http://www.essi.upc.edu/~ccares/papers/istarmlRefGuide.pdf.

[20] Hypertext Preprocessor PHP, http://www.php.net .

[21] PHP DOM XML, http://php.net/manual/en/book.dom.php .

[22] F. Dalpiaz. "Teaching Goal Modeling in Undergraduate Education", in *Proc. iStarT*, 2015, pp1-6.

[23] J. P. Carvallo. "Teaching Information Systems: an *i**-based approach", in *Proc. iStarT*, 2015, pp.7-12.

[24] E.O. Svee, J. Zdravkovic. " iStar Instruction in Mixed Student Cohort Environments", in *Proc. iStarT*, 2015, pp19-24.

[25] E. Yu, L. Lessard, Z. Babar, S. Nalchigar and J. Horkoff. "Teaching i* Alongside a Contrasting Modeling Framework", in *Proc. iStarT*, 2015, pp25-30.

[26] Z. Babar, S. Nalchigar, L. Lessard, J. Horkoff and E. Yu. "Instructional Experiences with Modeling and Analysis using the i* Framework", in *Proc. iStarT*, 2015, pp31-36.

[27] E. Paja, J. Horkoff and J. Mylopoulos. "The importance of teaching goal-oriented analysis techniques: an experience report", in *Proc. iStarT*, 2015, pp37-42.

[28] A. Bennaceur, J. Lockerbie and J. Horkoff. "On the Learnability of i*: Experiences from a New Teacher", in *Proc. iStarT*, 2015, pp43-48.

[29] A.P.A. Oliveira, V.M.B. Werneck, J. C.S.P. Leite and L.M. Cysneiros. "The Monopoly Game to Teach ERi*c - Intentional Requirements Engineering", in *Proc. iStarT*, 2015, pp49-54.

[30] C. Cares, X. Franch, L. Lopez and J. Marco. "Definition and Uses of the *i** Metamodel", in *Proc. iStar10*, 2010, pp20-24.

[31] J. Mylopoulos, M. Kolp and J. Castro. “UML for Agent Oriented Software Development: The Tropos Proposal”, volume 2185 of Lecture Notes in Computer Science, 2001, pages 422–441.

[32] P. Giorgini, J. Mylopoulos, and R. Sebastiani. Goal-oriented requirements analysis and reasoning in the Tropos methodology. Engineering Applications of Artificial Intelligence, 18(2):159–171, 2005.

[33] A. Fuxman, M. Pistore, J. Mylopoulos and P. Traverso. “Model checking early requirements specifications in Tropos”, in Proceedings of the Fifth IEEE International Symposium on Requirements Engineering,, pages 174–181. IEEE, 2001.

[34] Canada University of Toronto. GRL - Goal-oriented Requirement Language. http://www.cs.toronto.edu/km/GRL/, 2015.

[35] Hajer Mejri. “Insights on How to Enhance the Detection of Modeling Errors by *iStar* Novice Learners”, in American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), 2017, Volume 32, No 1, pp. 160-167.

[36] HiME User Manual. Available at:

http://www.upc.edu/gessi/istar/tools/hime/download/Hime.UserGuide.pdf

[37] V. Abdelzad, D. Amyot, S. A. Alwidian and T. C. Lethbridge. “A Textual Syntax with Tool Support for the Goal-oriented Requirement Language,” in *Proc. iStar*, 2015, pp.61-66.

[38] Hajer Mejri and Pauline N. Kawamoto: “Development and Evaluation of an Online Assistant for Helping Novice Learners Correct Defects in i* Models”; 2016 Convention Record of the Shin-Etsu Chapter of the Institute of Electronics, Information, and Communication Engineers, IEEE Shin-etsu Poster Session, P2-7, October 8, 2016, 165.

[39] *i** Models Design Quality and Contents Reviewer Toolkit:

http://lang.cs.shinshu-u.ac.jp/p/hajer/Toolkit/uploader.html

Appendix A: Source code and screenshot of the toolkit homepage

A.1 Toolkit overview

Our proposed toolkit includes two tools (*i**Check and GENERAT*i**ON) that enable the *i** goal modeling's novice learners to check and review their created *i** requirements diagrams' both layers namely the syntactical and contents qualities using the Internet service. Each of aforementioned tools allows its users in general, either beginners or experts, to connect to the server through a standard browser and proceed with their desired review type. Figure A.1 shows the homepage for the latest version of our toolkit.

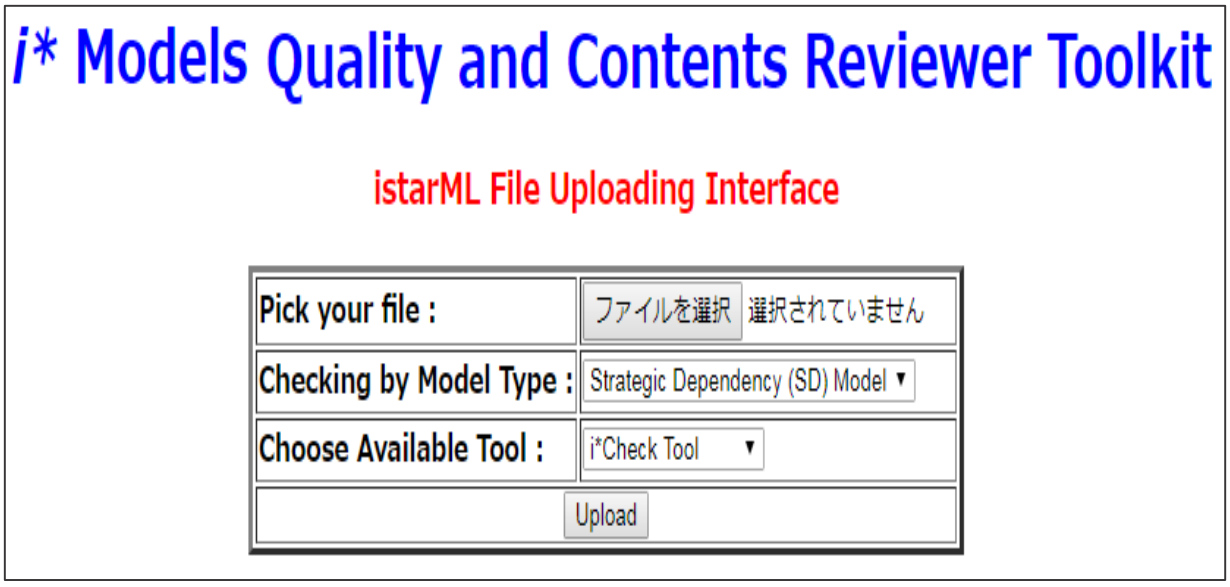


Figure A.1: Toolkit homepage interface [39]

The toolkit homepage is created by means using HTML language and it is named as “uploader.html”. Mainly, it includes a form that allows the user to browse to his desired model file which is in iStarML format (either for model quality checking purpose or for model content validation), choose the type of his model (SD or SR model), indicate the tool that he wants to use (i.e. the type of review he wants to perform on the model) and finally he submits the iStarML file. Once clicked on, the upload button will send the submitted form information to a PHP file with the name (fileUpload.php). This PHP file does the necessary checks that are indispensable for a sound and correct iStarML uploading to a chosen tool like verifying the extension of the uploaded file (only files with the “istarml” extension and format are supported and accepted by the toolkit and in case of different extension, an error message would appear on the screen and no uploading is performed), its size and if the file already exists (if it exists, a “-” and the next available number will be added to the file name. After performing all the aforementioned checks the file will be uploaded by moving and redirecting it to a specific PHP file for a subsequent model review.

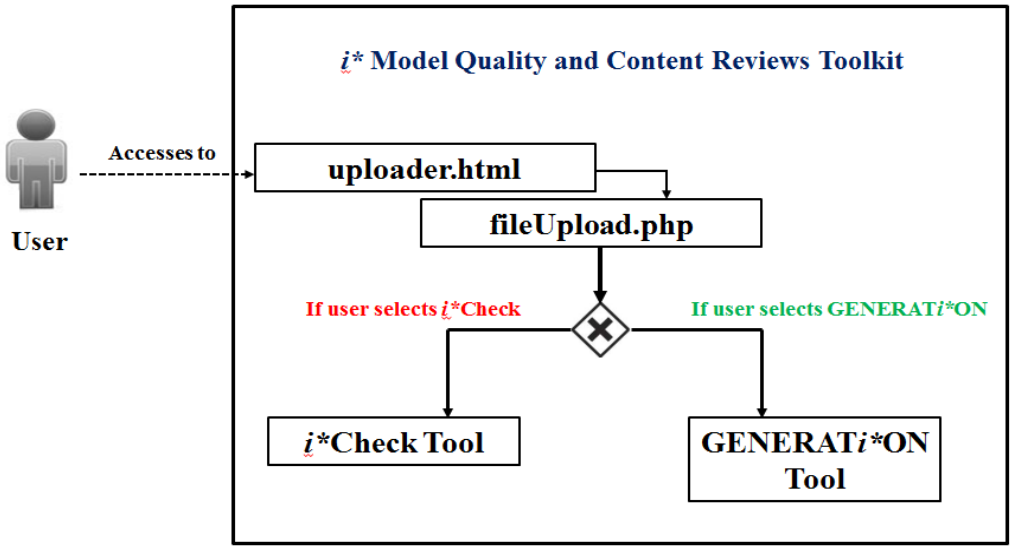


Figure A.2 Overview of the i* model quality and content reviews toolkit architecture

A.2 Source codes

A.2.1 uploader file code (HTML)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Welcome to our i* models uploading web page!</title>
</head>
<body>
   <center>
```

```
<h1><font color = "BLUE"><i>i*</i> Models Quality and Contents Reviewer Toolkit</font></h1>
            <h3><font color = "RED">istarML File Uploading Interface</font></h3>

<form method="post" action="fileUpload.php" enctype="multipart/form-data">
  <table border="3">

    <tr>
      <td><b>Pick your file :</b></td>
      <td><input type="file" name="userfile" /></td>
    </tr>
    <tr>
      <td><b>Checking by Model Type :</b></td>
                                    <td>
                                       <select name="selectType" onchange="redirect()">
          <option value="sd">Strategic Dependency (SD) Model</option>
          <option selected="selected" value="sr">Strategic Rationale (SR) Model</option>
        </select>
                                    </td>
    </tr>
                                <tr>
      <td><b>Choose Available Tool :</b></td>
                                    <td>
                                       <select name="selectTool" onchange="redirect()">
          <option value="i*check" selected="selected">i*Check Tool</option>
          <option  value="generati*on">Generati*on Tool</option>
        </select>
                                    </td>
    </tr>


    <tr>
      <td colspan="2" align="center"><input type="submit" value="Upload" /></td>
  </table>
</form>

</center>
</body>
</html>
```

A.2.2 fileUpload.php code (PHP)

```
<?php

  if(isset($_FILES['userfile'])){
    $errors= array();
    $file_name = $_FILES['userfile']['name'];
    $file_size =$_FILES['userfile']['size'];
    $file_tmp =$_FILES['userfile']['tmp_name'];
    $file_type=$_FILES['userfile']['type'];
    $file_ext=strtolower(end(explode('.',$_FILES['userfile']['name'])));
```

```
        $select= $_REQUEST['selectType'];
        $select1= $_REQUEST['selectTool'];

    $extension= array("istarml");

        //cheching for any file errors
    $errors_Found=FALSE;
        //to check the extension of the uploaded file
    if(in_array($file_ext,$extension)=== false)
        {
            $errors_Found=True;

            echo "extension not allowed, please choose an istarml file.";
    }
    //to check the size of the uploaded file
    if($file_size > 5000000)
        {
          $errors_Found=True;
      echo 'File size must not exceed be 5 MB';
    }

   //to check if the file already exists, if so it adds - and the next available number
        $i = 0;
    $parts = pathinfo($file_name);
        echo $parts["filename"];
    while (file_exists("upload/" . $file_name))
        {
     $i++;
     $file_name = $parts["filename"] . "-" . $i . "." . $file_ext;
    }


        if ($select1 == 'i*check')
        {
          move_uploaded_file($file_tmp,"upload/" . $file_name);
        }
        elseif($select1 == 'generati*on')
        {
          move_uploaded_file($file_tmp,"UploadedFiles/" . $file_name);
    }

function redirect($where)
  {
     header("Location: $where");
  }
  if ($select1 == 'i*check')
        {

          if ($select == 'sd')
              {
      redirect("SDCheck.php?filename=$file_name&checkType=$select");
```

```
        }
                elseif($select == 'sr')
                {
        redirect("try.php?filename=$file_name&checkType=$select");
        }
            }
    elseif($select1 == 'generati*on')
            {

                redirect("ParseForGeneration00++.php?filename=$file_name&checkType=$select");

  }

?>
```

Appendix B: Source code and screenshots of the i*Check tool

This appendix contains the source code and the returned feedback screenshots that are relative to the *i**Check tool which was developed with the purpose of detecting defects in the submitted *i** goal models of the novice learners and then guiding them to introduce correction by offering hints and suggestions on possible and potential solutions for each particular error caught (it suggest a set of actions to do in order to improve the conceptual model). Put differently, *i**Check returns two types of feedback: textual feedback which describes the detected defect and GIF animation which proposes corrective steps to follow.

We used PHP as main programming language, JavaScript was used to define the showing and hiding (showHide) function for the GIF animations feedback. We used CSS to describe how HTML elements are displayed on screen. Each uploaded (submitted for *i**Check) iStarML file will be stored in a folder in server called "upload".

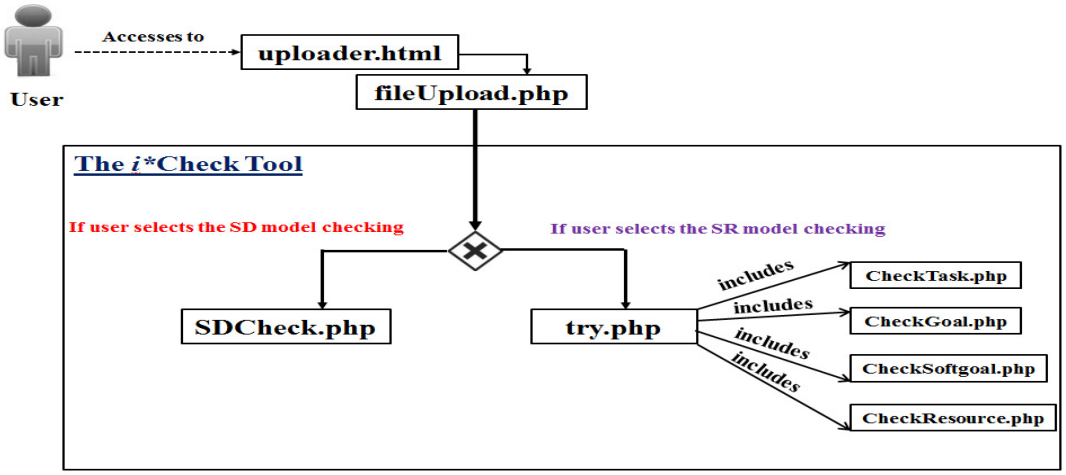


Figure B.1: Description of *iCheck tool's components for SD and SR checking respectively.**

B.1 SD Checking

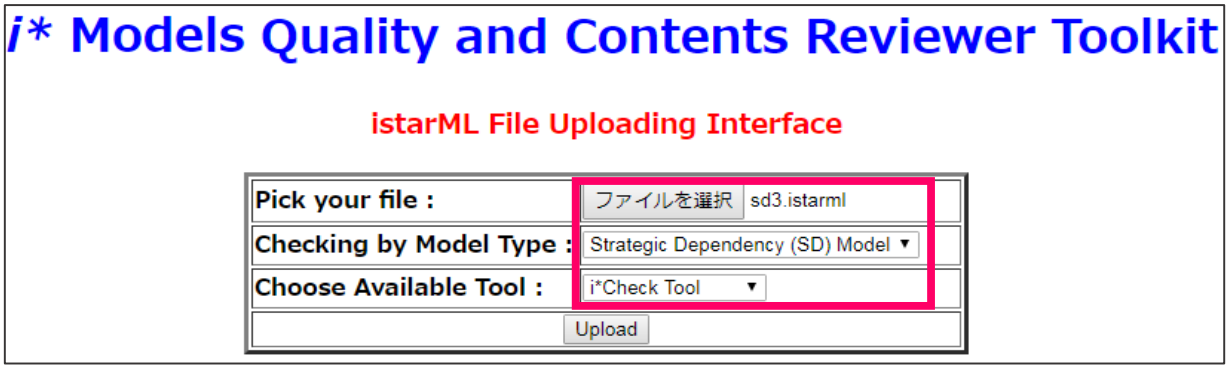


Figure B.2: An example of selecting an SD model "sd3" to submit to the *iCheck prototype**

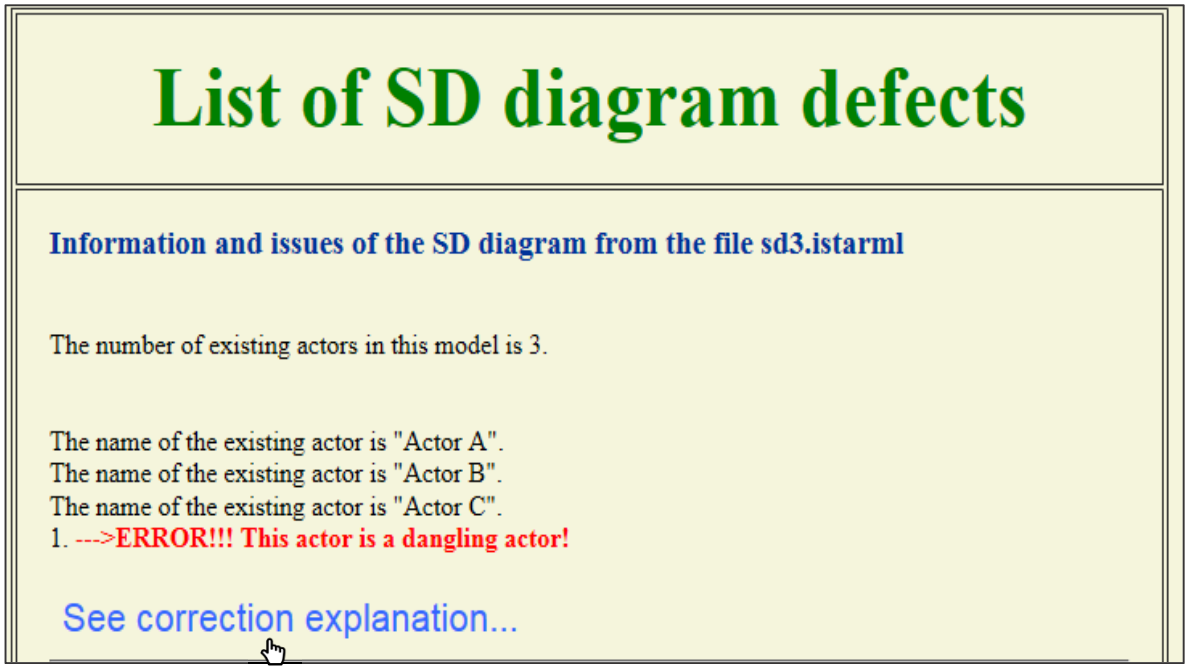


Figure B.3: Example of feedback returned from *iCheck showing an error in a SD model**

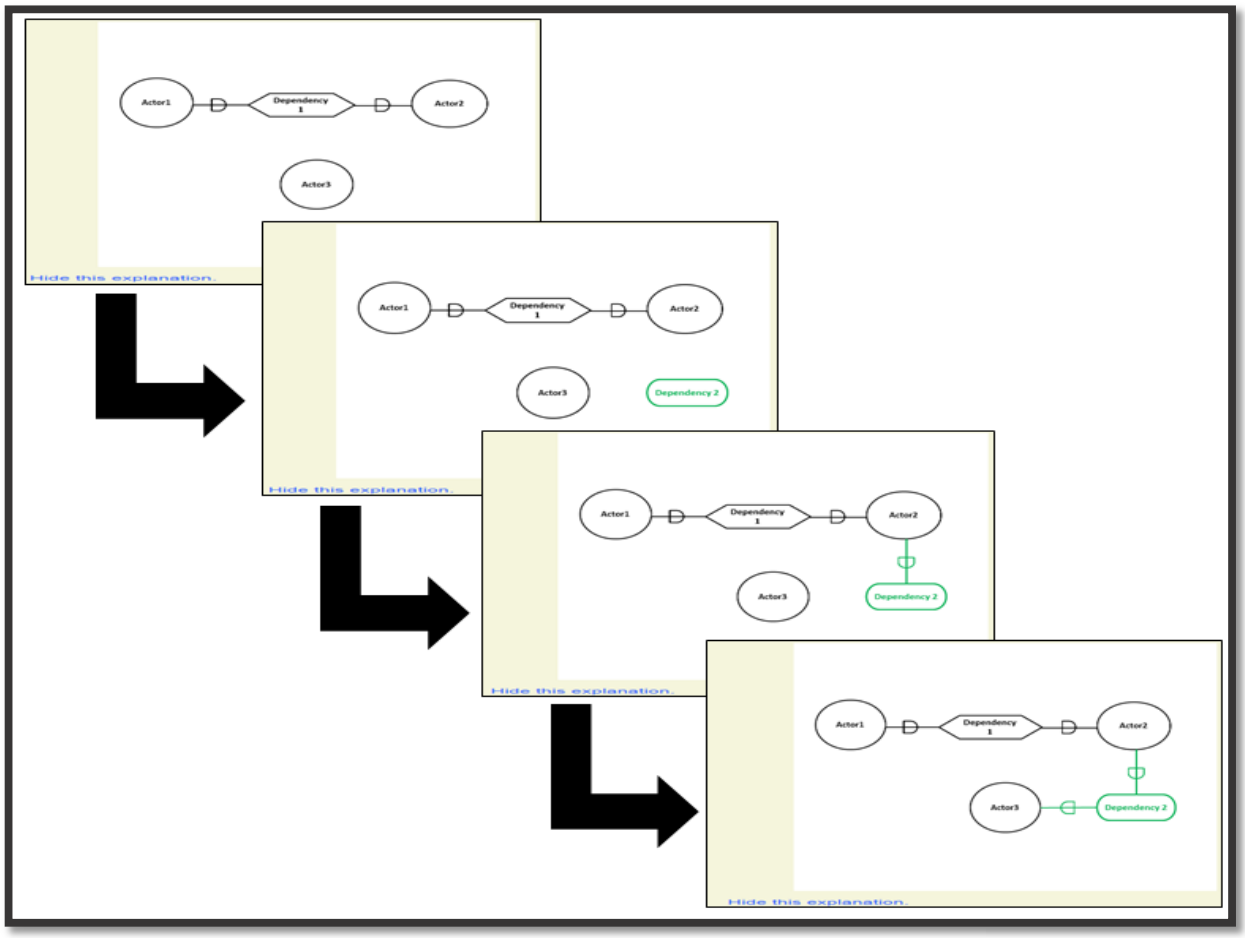


Figure B.4: The GIF animation corrective steps to solve the error detected in sd3.istarml file

B.1.1 SD checking part source code

SDCheck.php file source code

```
<script language="javascript" type="text/javascript">
function showHide(shID) {
  if (document.getElementById(shID)) {
    if (document.getElementById(shID+'-show').style.display != 'none') {
      document.getElementById(shID+'-show').style.display = 'none';
      document.getElementById(shID).style.display = 'block';
    }
    else {
      document.getElementById(shID+'-show').style.display = 'inline';
      document.getElementById(shID).style.display = 'none';
    }
  }
}
```

```
</script>
<style type="text/css">
  /* This CSS is just for presentational purposes. */
  body {
    font-size: 62.5%;
    background-color: beige; }
  #wrap {
    font: 1.4em/1.3 Arial, Helvetica, sans-serif;
    width: 30em;
    margin: 0 auto;
    padding: 0em;
    background-color: transparent; }

  /* This CSS is used for the Show/Hide functionality. */
  .more {
    display: none;
    border-top: 1px solid #666;
    border-bottom: 1px solid #666; }
  a.showLink, a.hideLink {
    text-decoration: none;
    color: #36f;
    padding-left: 8px;
    background: transparent url(down.gif) no-repeat left; }
  a.hideLink {
    background: transparent url(up.gif) no-repeat left; }
  a.showLink:hover, a.hideLink:hover {
    border-bottom: 1px dotted #36f; }
</style>


<?php
$dom = new DOMDocument();
$dom->preserveWhiteSpace = FALSE;
$name= $_REQUEST['filename'];
$type= $_REQUEST['checkType'];
$dom->load("upload/$name");
$xpath = new DOMXpath($dom);

$actors = $xpath->query('//actor');

function getDepParentName ( $dependency )
{
        $Parent= $dependency->parentNode;
        $ParentName= $Parent->getAttribute('name');

        return $ParentName;

}

function ActorIDExists ( $id )
{
```

```
	global $actors;
	foreach ( $actors as $actor )
	{
		$identifier=$actor->getAttribute('id');
		if ( $identifier == $id )
		{
			return 1;
		}

	}
	return 0; //no actor in the specified list had that given $id

}

function Check_for_isolated_dep ( $dependency )
{
	global $actors;

	//if the dependency does not have a valid der AND valid dee (attribute aref exist and it has some
valid value) so we return 1, else we return 0.
	if (!$dependency->hasChildNodes())
	{
		return 1;
	}

	$listDers = $dependency->getElementsByTagName('depender');
  $listDees = $dependency->getElementsByTagName('dependee');
	$numb_Ders= $listDers->length;
	$numb_Dees= $listDees->length;
	if ($numb_Ders == 0 and $numb_Dees == 0)
	{
		return 1;
	}
	//First we checked the dependers list side
	foreach ($listDers as $der)
	{
		if ($der->hasAttributes())
    {
      $arefFound=FALSE;
      foreach ($der->attributes as $attr)
        {

          $namer = $attr->nodeName;
          $valuer = $attr->nodeValue;
            if ($namer=='aref')
            {
                $arefFound=true;
                                        if ($valuer!='')
                {
                                                $actorExists= ActorIDExists ( $valuer );
                                                if ( $actorExists)
```

```
                                                                {
                                                                        return 0;  // this dependency has AT
LEAST one VALID Der specification which means the dependency IS NOT ISOLATED!!!!
                                                                }
                                        }

                                                }
                                        }
                        }



    }

            //Second we check the dependees list side
            foreach ($listDees as $dee)
            {
                    if ($dee->hasAttributes())
        {
            $arefFound=FALSE;
            foreach ($dee->attributes as $attr)
                {

                    $namee = $attr->nodeName;
                    $valuee = $attr->nodeValue;
                            if ($namee=='aref')
                            {
                                    $arefFound=true;
                                                    if ($valuee!='')
                                    {
                                                                    $actorExists= ActorIDExists ( $valuee  );
                                                                    if ( $actorExists)
                                                                    {
                                                                            return 0;  // this dependency has AT
LEAST one VALID Dependee specification which means the dependency IS NOT ISOLATED!!!!
                                                                    }
                                        }

                                                }
                                        }
                        }



    }
return 1;
}

function display_SD(){

$dom = new DOMDocument();
```

```
$dom->preserveWhiteSpace = FALSE;
$name= $_REQUEST['filename'];
$dom->load("upload/$name");
$xpath = new DOMXpath($dom);


global $actors;
//$actors = $xpath->query('//actor');
$ielements = $xpath->query('//ielement');
$AllDependencies = $xpath->query('//dependency');
$Alldependees = $xpath->query('//dependee');
$Alldependers = $xpath->query('//depender');
$AllIntentionalLinks = $xpath->query('//ielementLink');
$diagramList= $dom->getElementsByTagName('diagram');
$diagram= $diagramList->item(0);
$diagramName = $diagram->getAttribute('name');
echo '<h3><font color="#003399">Information and issues of the SD diagram from the file
'.$diagramName.'</font></h3>',"<br />";

echo 'The number of existing actors in this model is '. $actors->length. '.' ,"<br />";
echo "<br />";

// It was a code to remove the unwanted elements. WE DIDN'T touch the source file!!!
$GraphicNodes = $dom->getElementsByTagName('graphic');

if ($GraphicNodes->length == 0)
{
        $done = True;  //the istarml File has no graphic nodes ()tags)
}
else
{
        $done = False;
}
   for (; !$done; )
   {


      $gnode=$GraphicNodes->item(0);


      $gnode->parentNode->removeChild($gnode);


      if ($GraphicNodes->length == 0)
      {
                  $done = True;  // no more graphic nodes left
            }


   }
```

```
$error_label=1;
//displaying the ordered list of errors

//Check #7
$error_number= 1; // it starts with 1 because it is used in counting the number of appearance of a certain error
if($actors->length==0 OR $actors->length==1 ) // if ($actors->length < 2)
{

  echo $error_label.'. <Font COLOR = RED><b>--->ERROR!!! There should be at least two actors in the i* model!! </b></Font>',"<br />";
  $error_label++;
          echo '<div id="wrap">';
     echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
     echo '<div id="example_'.$error_number.'" class="more">';
       echo '<p><center><img src="./solutionPattern7.gif" width = "500" height="800"></center></p>';
       echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink" onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
     echo '</div>';
   echo '</div>';

  $error_number++;
}

          echo "<br />";



//Check #3:
foreach ($actors as $actor)
{
          $acteurs= $actor->getAttribute('name');
          echo 'The name of the existing actor is "'. $acteurs.'".' ,"<br />";

          $IdA= $actor->getAttribute('id');
          // this loop is iterating to make sure that one actor appears at least in one dependency either as depender or as dependee!!

          $found = FALSE;
          foreach($Alldependers as $depender)
          {

      $dependerId= $depender->getAttribute('aref');


                    if ($IdA==$dependerId)
                    {
                              $found=True;

                    break;
```

```
                }
        }
        if (!$found)
        {
                foreach ($Alldependees as $dependee)
                {
                        $dependeeId= $dependee->getAttribute('aref');
                        if ($IdA==$dependeeId)
                        {
                           $found=True;
                           break;
                        }
                }
        }
        if (!$found) // if the actor id is still not found
        {

           echo $error_label.'. <Font COLOR = RED>---><b>ERROR!!! This actor is a dangling
actor!</b></Font>', "<br />";

                $error_label++; //increment the error_label
          echo '<div id="wrap">';
     echo '<p><a href="#" id="example2_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example2_'.$error_number.'\');return false;">See correction explanation...</a></p>';
     echo '<div id="example2_'.$error_number.'" class="more">';
      echo '<p><center><img src="./solutionPattern3.gif" width = "500" height="800"></center></p>';
      echo '<p><a href="#" id="example2_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example2_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
     echo '</div>';
     echo '</div>';

        $error_number++;

   }

}

        echo '<HR>';



if ( $AllDependencies->length!=0 ) //if the list of dependencies is not empty
{
   foreach($ielements as $ielement)
        {
           $nom = $ielement->getAttribute('name');
           //check if the ielement is not inside an actor boundary (dependency)
                $Parent=$ielement->parentNode;
                $ParentName=$Parent->nodeName;
                if ($ParentName!='boundary')
                {
```

```
if ($ielement->hasChildNodes())
    { //check if ielement has children called dependency

$dependencies=$ielement->getElementsByTagName('dependency'); //grab only the
dependency tagged children

if ($dependencies->length!=0)
  {  //if the ielement has children tagged as dependency
      foreach($dependencies as $dependency)
      { //get the  dependencies list one by one
          $depName = $ielement->getAttribute('name'); //go get the dependency
name
          echo 'this dependency is labeled as "'.$depName.'"', "<br />"; //display the
dependency name
    $dependers= $dependency->getElementsByTagName('depender');
  $dependees= $dependency->getElementsByTagName('dependee');
    $ders= $dependers->length;
    $dees=$dependees->length;
  echo 'number of dependers in this dependency is '.$ders,"<br />";
      echo 'number of dependees in this dependency is '.$dees,"<br />";


  //Check #4
    if ($ders>1 and $dees==0)
    {

        echo $error_label.'. <Font COLOR = RED><b> --->ERROR!!! Each
dependency must have exactly one depender and one dependee! <b></Font>',"<br />";
                                $error_label++;
                                echo '<div id="wrap">';
    echo '<p><a href="#" id="example3_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example3_'.$error_number.'\');return false;">See correction explanation...</a></p>';
    echo '<div id="example3_'.$error_number.'" class="more">';
    echo '<p><center><img src="./solutionPattern4.gif" width = "500"
height="800"></center></p>';
    echo '<p><a href="#" id="example3_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example3_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
    echo '</div>';
    echo '</div>';

    $error_number++;


    }
   //Check #8
    if ($ders ==1 and $dees==0)
                        {

        echo $error_label.'. <Font COLOR = RED><b>--->STOP!!! Each
dependency has to have one and only one depender AND one and only one dependee!!</b> </Font>',"<br
/>";
```

```
                                                    $error_label++;

                                                    echo '<div id="wrap">';
                    echo '<p><a href="#" id="example4_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example4_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                    echo '<div id="example4_'.$error_number.'" class="more">';
                    echo '<p><center><img src="./solutionPattern8.gif" width = "500"
height="800"></center></p>';
                    echo '<p><a href="#" id="example4_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example4_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                    echo '</div>';
                    echo '</div>';

                      $error_number++;

                    }


                    //Check #9
                    if ($ders >1 and $dees==1)
                                        {

                    echo $error_label.'. <Font COLOR = RED><b>--->ERROR!!! Two Dependency Links
outgoing from two Dependers should not be joined to target one Dependum </b></Font>',"<br />";
                                echo '<Font COLOR = GREEN >---> Only one link should be drawn on both
sides of a Dependum!!!</Font>',"<br />";

                                                    $error_label++;

                                                    echo '<div id="wrap">';
                    echo '<p><a href="#" id="example5_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example5_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                    echo '<div id="example5_'.$error_number.'" class="more">';
                    echo '<p><center><img src="./solutionPattern9.gif" width = "500"
height="800"></center></p>';
                    echo '<p><a href="#" id="example5_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example5_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                    echo '</div>';
                    echo '</div>';

                      $error_number++;

                    }
                                                //Check #10
                if ($ders == 1 and $dees>1)
                    {
                      echo $error_label.'. <Font COLOR = RED><b>--->ERROR!!! A Dependency
Link should not be split into two links from one Dependum to two Dependees! </b></Font>',"<br />";

                        $error_label++;
```

```
                                echo '<Font COLOR = GREEN >Only one link should be drawn on both sides
of a Dependum!!!</Font>',"<br />";
                                                        echo '<div id="wrap">';
                        echo '<p><a href="#" id="example6_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example6_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                        echo '<div id="example6_'.$error_number.'" class="more">';
                        echo '<p><center><img src="./solutionPattern10.gif" width = "500"
height="800"></center></p>';
                        echo '<p><a href="#" id="example6_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example6_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                        echo '</div>';
                        echo '</div>';

                                $error_number++;
                            }
                         }

                }
                    }


//this else is the else of the if ($ielement->hasChildNodes()) above
            else
                            {
                                echo $error_label.'. <Font COLOR = RED><b>--->ERROR! There exist some
disconnected elements in this model: </b></Font>'. $nom, '<br>';
                                    $error_label++;

                                    echo '<div id="wrap">';
                echo '<p><a href="#" id="example7_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example7_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                echo '<div id="example7_'.$error_number.'" class="more">';
                echo '<p><center><img src="./SD02V2.gif" width = "500" height="800"></center></p>';
                echo '<p><a href="#" id="example7_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example7_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                echo '</div>';
                echo '</div>';

                        $error_number++;

            }
            echo "<br />";

        }

    }
}



//Check #1  MODEL contains Actors ONLY!!!
```

```
else  //this else is the else of the if ($Alldependencies->length!=0) above
{
// I need to check if there is ielement


if  ( $ielements->length==0)
{
echo $error_label.'. <Font COLOR = RED> --->ERROR! This model does not contain any dependency
binding the actors! </Font>'  ,"<br />";
$error_label++;
echo '<div id="wrap">';
echo '<p><a href="#" id="example8_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example8_'.$error_number.'\');return false;">See correction explanation...</a></p>';
echo '<div id="example8_'.$error_number.'" class="more">';
echo '<p><center><img src="./solutionPattern1.gif" width = "500" height="800"></center></p>';
echo '<p><a href="#" id="example8_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example8_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
echo '</div>';
echo '</div>';

$error_number++;
}
}

//check #5: The dependum is absent
foreach ( $AllDependencies as $dependency)
{
   $DepChildren=$dependency->childNodes;
          if ($DepChildren->length ==2)
          {
             $dependerList= $dependency->getElementsByTagName('depender');
             $dependeeList= $dependency->getElementsByTagName('dependee');
             foreach ($dependerList as $depender)
                    {
                $der=$depender->getAttribute('aref');
                $Found=FALSE;

          foreach ($actors as $actor)
                          {
                        $id = $actor->getAttribute('id');
                        if ($id==$der)
                                    {
                                $Found=True;
                                $nameDer = $actor->getAttribute('name');
                                break;
                              }
             }
          }
          foreach ($dependeeList as $dependee)
                    {
             $dee=$dependee->getAttribute('aref');
```

```
                $Found=FALSE;

                foreach ($actors as $actor)
                        {
                        $id = $actor->getAttribute('id');
                        if ($id==$dee)
                                {
                                $Found=True;
                                $nameDee = $actor->getAttribute('name');
                                break;
                                }
                        }
                }
            }

        }
    $parent= $dependency->parentNode;
    $ParentName=$parent->nodeName;
    if ($ParentName != 'ielement')
        {

        echo $error_label. '. <Font COLOR = RED> --->ERROR! The dependum is absent between
</Font>"'.$nameDer. '"<Font COLOR = RED> and </Font>"'.$nameDee.'"',"<br />";
        $error_label++;

        echo '<div id="wrap">';
    echo '<p><a href="#" id="example9_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example9_'.$error_number.'\');return false;">See correction explanation...</a></p>';
    echo '<div id="example9_'.$error_number.'" class="more">';
    echo '<p><center><img src="./solutionPattern5.gif" width = "500" height="800"></center></p>';
    echo '<p><a href="#" id="example9_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example9_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
    echo '</div>';
    echo '</div>';

        $error_number++;
        }

}

//Check for Disconnection #2 EVERY ELEMENT IS ISOLATED

//if AT LEAST one actor AND ielement exist in the model
if ($ielements->length>0 and $actors->length>0)
{
$non_isolated_dependency_found = FALSE;
foreach ( $AllDependencies as $dependency)
{



$status = Check_for_isolated_dep ( $dependency );
```

```
   if ( $status == 0)
   {
      $non_isolated_dependency_found = true;
      break;

   }

}
if (!$non_isolated_dependency_found)
{
      echo $error_label.'. <Font COLOR = RED> --->ERROR! All elements of model are completely
disconnected! </Font>',"<br />";
                  $error_label++;

                  echo '<div id="wrap">';
      echo '<p><a href="#" id="example10_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example10_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
      echo '<div id="example10_'.$error_number.'" class="more">';
      echo '<p><center><img src="./solutionPattern2.gif" width = "500" height="800"></center></p>';
      echo '<p><a href="#" id="example10_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example10_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
      echo '</div>';
      echo '</div>';

            $error_number++;

}

}
echo '<HR>';

//check #6
foreach ($AllIntentionalLinks as $Link)
{
         $LinkParent= $Link->parentNode;
         $done=FALSE;
         $found=False;
         while (!$done && !$found)
         {
                  $Name=$LinkParent->nodeName;
            if ($Name == 'boundary')
                  {
                     $found =True;
            }
            else
                  {
                     if ($Name== 'diagram')
                           {
                              $done = True;
                           }
```

```
                    else
                        {
                            $LinkParent= $LinkParent->parentNode;
                        }
                }
        }
        if (!$found)
        {
                echo $error_label.'. <Font COLOR = RED> --->ERROR! There should not be any
decomposition outside the actor boundary! </Font>', "<br />";
                $error_label++;
                echo '<div id="wrap">';
    echo '<p><a href="#" id="example11_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example11_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
    echo '<div id="example11_'.$error_number.'" class="more">';
    echo '<p><center><img src="./solutionPattern6.gif" width = "500" height="800"></center></p>';
    echo '<p><a href="#" id="example11_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example11_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
    echo '</div>';
    echo '</div>';

        $error_number++;

        }
}

}// End of display_SD();
echo '<TABLE border = 1 cellpadding = 20>';
echo '<TR>';
echo '<TD> <center><b><font size = 14 color="GREEN">List of SD diagram
defects</font></b></center></TD>';
echo '</TR>';

echo '<TD valign=top>',display_SD();
echo '</TD>';
?>
```

B.2 SR Checking

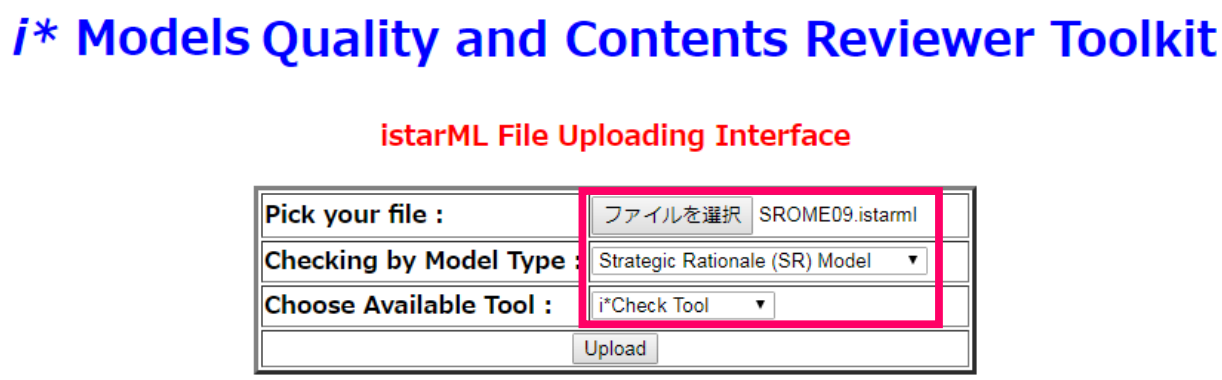


Figure B.5: An example of selecting an SR model called "SROME09" to submit to the *iCheck prototype**

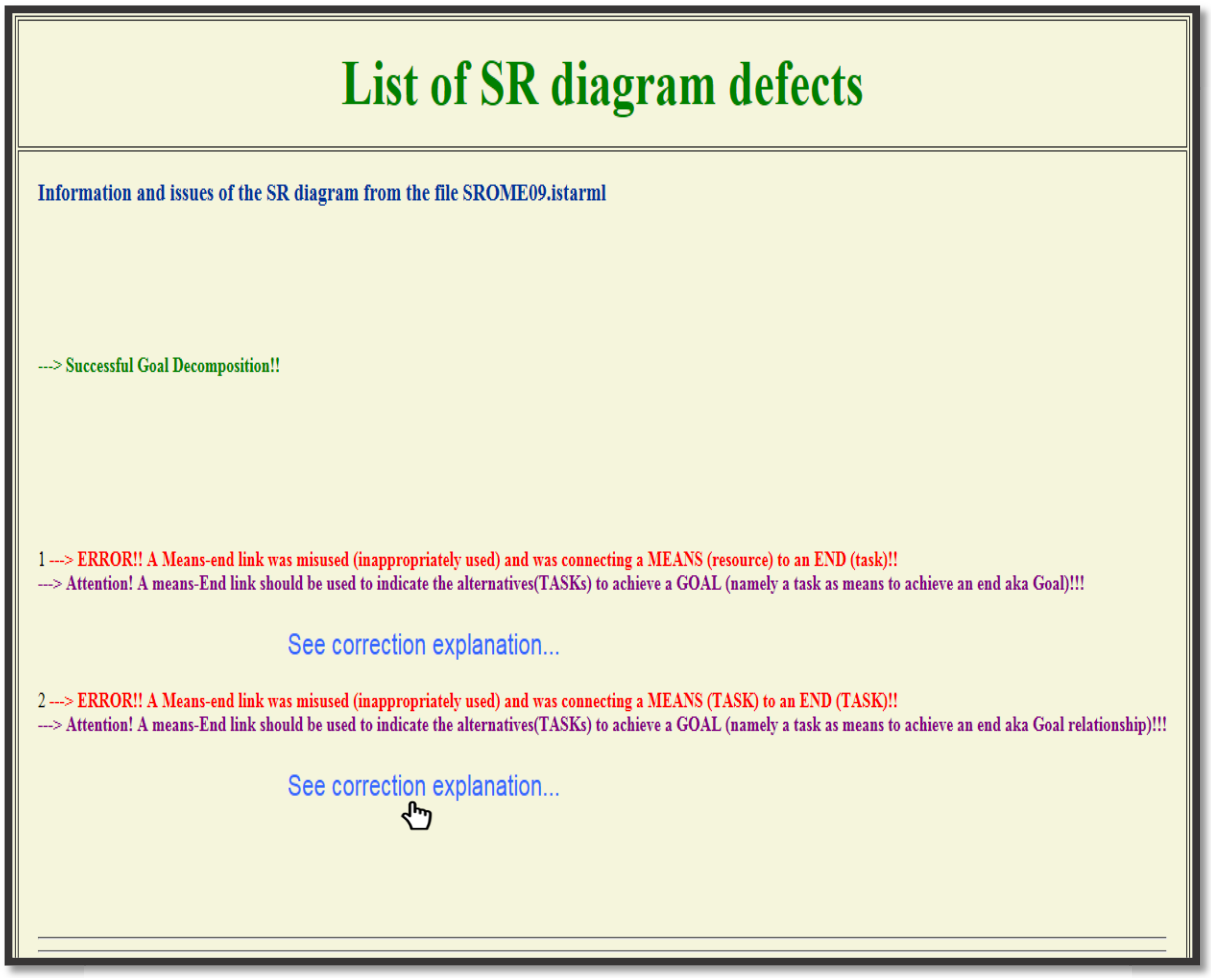


Figure B.6: The feedback returned from *iCheck showing a list of errors in SR model**

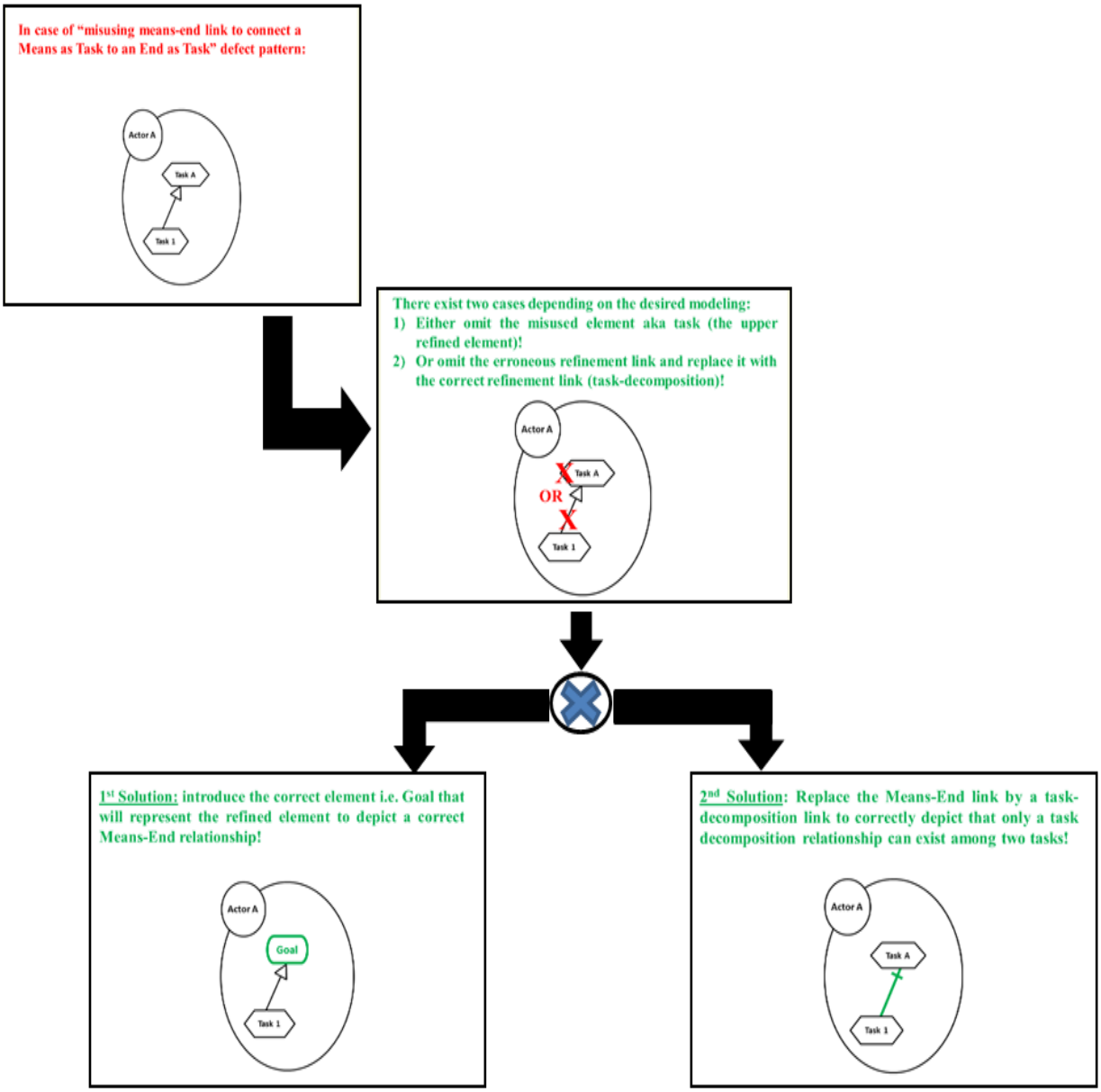


Figure B.7: Example of GIF animation offering steps to correct a specific SR error

B.2.1 SR part checking code source

try.php file source code

```
<script language="javascript" type="text/javascript">
function showHide(shID) {
  if (document.getElementById(shID)) {
    if (document.getElementById(shID+'-show').style.display != 'none') {
```

```
        document.getElementById(shID+'-show').style.display = 'none';
        document.getElementById(shID).style.display = 'block';
      }
      else {
        document.getElementById(shID+'-show').style.display = 'inline';
        document.getElementById(shID).style.display = 'none';
      }
    }
}
</script>
<style type="text/css">
  /* This CSS is just for presentational purposes. */
  body {
    font-size: 62.5%;
    background-color: beige; }

  #wrap {
    font: 1.4em/1.3 Arial, Helvetica, sans-serif;
    width: 30em;
    margin: 0 auto;
    padding: 0em;
    background-color: transparent; }

  /* This CSS is used for the Show/Hide functionality. */
  .more {
    display: none;
    border-top: 1px solid #666;
    border-bottom: 1px solid #666; }
  a.showLink, a.hideLink {
    text-decoration: none;
    color: #36f;
    padding-left: 8px;
    background: transparent url(down.gif) no-repeat left; }
  a.hideLink {
    background: transparent url(up.gif) no-repeat left; }
  a.showLink:hover, a.hideLink:hover {
    border-bottom: 1px dotted #36f; }
</style>

<?php
//include 'SDCheck.php';  //try before server  and change the file name in this file copy in server to contain
the same SRFileName


include 'CheckTask.php';
include 'CheckGoal.php';
include 'CheckSoftgoal.php';
include 'CheckResource.php';
function display_SR()
{
$dom = new DOMDocument();
```

```
$dom->preserveWhiteSpace = False;
//$dom->load('SRFileName.istarml');
$name= $_REQUEST['filename'];
$dom->load("upload/$name");
$xpath = new DOMXpath($dom);
$Allactors = $xpath->query('//actor');
$Allboundaries = $xpath->query('//boundary');
$Allielements = $xpath->query('//ielement');  //all of the ielements
$diagramList=$dom->getElementsByTagName('diagram');
$diagram=$diagramList->item(0);
$diagramName = $diagram->getAttribute('name');
$Alldependencies= $xpath->query('//dependency');
$ieLinksList= $xpath->query('//ielementLink');


echo '<h3><font color="#003399">Information and issues of the SR diagram from the file
'.$diagramName.'</font></h3>',"<br />";


// It was a code to remove the unwanted elements. WE DIDN'T touch the source file!!!
$GraphicNodes = $dom->getElementsByTagName('graphic');

if ($GraphicNodes->length == 0)
{
        $done = True;  //the istarml File has no graphic nodes ()tags)
}
else
{
        $done = False;
}
    for (; !$done; )
                {


     $gnode=$GraphicNodes->item(0);


        $gnode->parentNode->removeChild($gnode);


        if ($GraphicNodes->length == 0)
        {
                                $done = True;  // no more graphic nodes left
                        }


    }
```

```
//DEVIATION aka Divergence from purpose text + GIF animation
$error_number= 1;
$error_label= 1;
//From check 7~12
foreach ($Allactors as $actor)
{

   if ($actor->hasChildNodes())
   {
            $bound=$actor->firstChild;
      if($bound->hasChildNodes())
            {
               $boundaryKids=$bound->childNodes;

               foreach ($boundaryKids as $ielement)
               {
            $ielementType = $ielement->getAttribute('type');
                  $intentional= $ielement->getAttribute('name');
                  $id=$ielement->getAttribute('id');

                  if ($ielementType=='task')
                  {
                        // echo 'This internal element is a '. $ielementType. ' and it is labeled as "'.
          $intentional. "" ,"<br />";
               list ($label, $number)=CheckTask($ielement, $ielementType, $intentional, $id, $bound,
$error_label, $error_number );
                     $error_label=$label;
                                    $error_number=$number;
                  }
                                 echo "<br />";
                                 if ($ielementType=='goal')
                                 {
                                  list ($label, $number)=CheckGoal($ielement, $ielementType,
$intentional, $id, $bound, $error_label, $error_number );
                     $error_label=$label;
                           $error_number=$number;

            }
                                 echo "<br />";
                                 if ($ielementType=='softgoal')
                                 {
                                 list ($label, $number)=CheckSG($ielement, $ielementType, $intentional,
$id, $bound, $error_label, $error_number );
                     $error_label=$label;
                           $error_number=$number;

                                 }
                                 echo "<br />";
                                 if ($ielementType=='resource')
                                 {
```

```
                                        list ($label, $number)=CheckResource($ielement, $ielementType,
$intentional, $id, $bound, $error_label, $error_number );
                    $error_label=$label;
                                $error_number=$number;

            }
                                        echo "<br />";

        }


            }

         }

}

//Check#1  Checking if a dependency link is drawn or used inside an actor boundary between internal
elements

foreach ($Allboundaries as $bound)
{

   if($bound->hasChildNodes())
          {
             $boundaryKids=$bound->childNodes;
          //echo $boundaryKids->length, "<br />";
             if ($boundaryKids->length>0)
             {
                foreach ($boundaryKids as $child)
                {
                        $childName=$child->getAttribute('name');
                            //echo $childName;
                   if ($child->hasChildNodes())
                   {
                      $childKids=$child->childNodes;
                                 if ($childKids->length>0)
                                 {
                                          foreach ($childKids as $kid)
                                          {
                                             if ($kid->nodeName=='dependency')
                                                    {
                                                       echo $childName, "<br />";
                                                       echo $error_number.'<font color = "RED">---
><b>Error!!! Dependency links cannot be used to connect two internal intentional elements  inside the
same actor!!</b></font>', "<br />";
                                 echo '<font color = "RED"><b>--->Dependency link can be used to connect an
internal element to a dependum!!!</b></font>', "<br />";
                                             echo '<div id="wrap">';
                          echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
```

```
                        echo '<div id="example_'.$error_number.'" class="more">';
                        echo '<p><center><img src="./GifSR01.gif" width = "500"
height="800"></center></p>';
                                    echo '<p><a href="#" id="example_'.$error_number.'-hide"
class="hideLink" onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this
explanation.</a></p>';
                        echo '</div>';
                        echo '</div>';
                                $error_number++;
                                                        }
                                        }


                                    }

            }

         }

      }
   }
}
  echo '<br />';
  echo '<HR>';

echo '<HR>';


//OLD Check #1 Checking if a dependency link is drawn or used inside an actor boundary between
internal elements
foreach ($Allboundaries as $border)
{
          $ieLinks = $border->getElementsByTagName('ielementLink');
   foreach($ieLinks as $lien)
          {
      $LinkType = $lien->getAttribute('type');
            if($LinkType == "dependency")
            {
                    echo $error_number.'<Font COLOR = RED><b> ---> ERROR!! A dependency link
should not exist inside an actor boundary!</b></Font>',"<br />";
                            echo '<div id="wrap">';
         echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
         echo '<div id="example_'.$error_number.'" class="more">';
         echo '<p><center><img src="./GifSR01.gif" width = "500" height="800"></center></p>';
                            echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
         echo '</div>';
         echo '</div>';
                $error_number++;
                }
```

```
        }

}


// CHECK #2: checking for actor existence inside another actor
foreach ($Allactors as $actor)
{

   if ($actor->hasChildNodes())
   {
            $bound=$actor->firstChild;
      if($bound->hasChildNodes())
            {
         $ElementsWithActorTag=$bound->getElementsByTagName('actor');

          // get the list of kids by actor name tag while it should not have ANY
              if ($ElementsWithActorTag->length!=0)
              {
                 foreach ($ElementsWithActorTag as $withinActor)
                              {
                    $nom= $withinActor->getAttribute('name');
                    echo $nom ,"<br />";
                              }
                 echo $error_number.'<font color="RED"><b> ---> ERROR!! An actor should not be
included within (inside) another actor!</b></font>',"<br />";
                 echo '<div id="wrap">';
            echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
            echo '<div id="example_'.$error_number.'" class="more">';
            echo '<p><center><img src="./GifSR02V2.gif" width = "500" height="800"></center></p>';
            echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
            echo '</div>';
            echo '</div>';
                                    $error_number++;


                           }

      }

   }

}


//CHECK #3: CHECK if the dependum is not connected to the right internal element
foreach ($Alldependencies as $dependency)
{
```

```
$dependerList= $dependency->getElementsByTagName('depender');
$dependeeList= $dependency->getElementsByTagName('dependee');
        $dependencyIElementParent = $dependency->parentNode;
$DepName= $dependencyIElementParent->getAttribute('name');

foreach ($dependerList as $der)
        { // get the aref of the current $der
   if ($der->hasAttribute('aref'))
                {
            $arefer=$der->getAttribute('aref');


         // go and find the actor id corresponding to the aref and check if the actor has internal ielements
            $Found=FALSE;
            foreach ($Allactors as $actor)
                    {
                $id = $actor->getAttribute('id');
                if ($id==$arefer)
                        {
                    $Found=True;
                    $nameDer = $actor->getAttribute('name');

                    $numberIE=0;
                  if ($actor->hasChildNodes())
                            {
                    $border=$actor->firstChild;
                    if ($border->hasChildNodes())
                                {
                        $internalElements=$border->childNodes;
                        $numberIE=$internalElements->length;

                    }
                  }

                  if ($numberIE != 0)
                            {
                    if (!$der->hasAttribute('iref'))
                                {
                                    echo $nameDer, "<br />";
                                        echo $DepName, "<br />";
                               echo $error_number.'<font color = "RED"><b>--->ERROR! The depender side of
this dependency is not connected to the right internal element</b></font>', "<br />";
                                        echo '<div id="wrap">';
                    echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                    echo '<div id="example_'.$error_number.'" class="more">';
                    echo '<p><center><img src="./GifSR03.gif" width = "500"
height="800"></center></p>';
                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                    echo '</div>';
```

```
                    echo '</div>';
                                        $error_number++;

                                                    }
                        else
                        {

                            $irefer=$der->getAttribute('iref');
                    //go check the internal elements which matches the corresponding iref
                            $Found=FALSE;
                            foreach ($internalElements as $ielement)
                                                            {
                                                $idElement = $ielement->getAttribute('id');


                        if ($idElement==$irefer)
                                {
                                        $Found=True;

                        }
                            }

                            if (!$Found)
                                                            {
                                        echo '<font color = "RED"><b>--->the iref in the depender is not matching
any internal element id! </b></font>', "<br />";// BE RIGHT BACK
                            }
                    }
                        }

                    }
                }
            }

   }

echo "<br />";
echo "<br />";

   foreach ($dependeeList as $dee)
          { // get the aref of the current $der
      if ($dee->hasAttribute('aref'))
                    {
                $arefee=$dee->getAttribute('aref');


                // go and find the actor id corresponding to the aref and check if the actor has internal
ielements
                $Found=FALSE;
                foreach ($Allactors as $actor)
                        {
```

```
                $id = $actor->getAttribute('id');
                if ($id==$arefee)
                        {
                    $Found=True;
                    $nameDee = $actor->getAttribute('name');

                    $numberIE=0;
                  if ($actor->hasChildNodes())
                                {
                        $border=$actor->firstChild;
                        if ($border->hasChildNodes())
                                        {
                                $internalElements=$border->childNodes;
                                $numberIE=$internalElements->length;

                        }
                  }


                  if ($numberIE != 0)
                                {


            if (!$dee->hasAttribute('iref'))
                                                {
                                                   echo $nameDee, "<br />";
                                                        echo $DepName, "<br />";
                        echo $error_number.'<font color = "RED">---><b>ERROR! The dependee side of
this dependency is not connected to the right internal element!</b></font>', "<br />";
                                                echo '<div id="wrap">';
                    echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                    echo '<div id="example_'.$error_number.'" class="more">';
                    echo '<p><center><img src="./GifSR0302.gif" width = "500"
height="800"></center></p>';
                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                    echo '</div>';
                    echo '</div>';
                                        $error_number++;


                                                }
                  else
                  {

                      $irefee=$dee->getAttribute('iref');
                   //go check the internal elements which matches the corresponding iref
                      $Found=FALSE;
                      foreach ($internalElements as $ielement)
```

```
                                                {
                                        $idElement = $ielement->getAttribute('id');


                        if ($idElement==$irefee)
                                {
                                        $Found=True;

                        }
                            }

                            if (!$Found)
                                                {
                                    echo '<font color = "RED"><b>--->the iref in the dependee is not
matching any internal element id! </b></font>', "<br />";// BE RIGHT BACK
                                }
                    }
                        }

                    }
                }
            }

   }

}
  echo '<br />';
  echo '<br />';
  echo '<br />';



// CHECK #4: Check for disconnected internal elements!!
foreach($Allboundaries as $border)
{
        $ieLinks = $border->getElementsByTagName('ielementLink');
        $insideElements= $border->getElementsByTagName('ielement');
        foreach($insideElements as $intentional)
        {
                $name= $intentional->getAttribute ('name');
        // does $intentional have any ielementLink children?
                $Links=$intentional->childNodes;
                $found = FALSE;
                if ($Links->length>0)
                {
                        foreach($Links as $liens)
                        {
                        // check if $intentional has any children as ielementLink (here it is source)
                                if ($liens->nodeName == 'ielementLink')
                                {
```

```
                                        $found = True;
                                        break;
                                }

                        }


                }
            if (!$found) // This means that the above checking fails
            {
                // check if $intentional is referenced in another ielement s ielementLink (here it is target)
                   $id = $intentional->getAttribute('id');
                        foreach ($ieLinks as $Lien)
                        {
                           $iref = $Lien->getAttribute('iref');

                           if ($id == $iref)
                                {
                                   $found=True;
                                   break;
                           }

                        }
                }
        if (!$found){
                           echo $error_number.'<font color="RED">---> ERROR!! The internal element
labeled as '.'<font color= "BLACK" > "' .$name. '"</font>'.' is not connected to another internal
element"!!</font>' ,"<br />";
                                echo '<font color="RED">---> ERROR!! This model is
incomplete!</font>' ,"<br />";
                                echo '<div id="wrap">';
            echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
            echo '<div id="example_'.$error_number.'" class="more">';
            echo '<p><center><img src="./GifSR04.gif" width = "500" height="800"></center></p>';
            echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
            echo '</div>';
                        $error_number++;

                                }
    }

}

echo "<HR>";


//CHECK #5: Check if an SR element is drawn outside the actor boundary
if($diagram->hasChildNodes())
```

```
{
        $diagramKids=$diagram->childNodes;
        if ($diagramKids->length>0)
        {
           echo 'number of diagram children  is: '. $diagramKids->length, "<br />";
           foreach ($diagramKids as $kid)
           {
              if ($kid->nodeName=='ielement')
              {


          $ielementType = $kid->getAttribute('type');
                $intentional= $kid->getAttribute('name');
                $id=$kid->getAttribute('id');

           if ($kid->hasChildNodes())
                {

                   $ielementsChildren=$kid->childNodes;

              $childrenByTag=$kid->getElementsByTagName('ielementLink');
                   if ($childrenByTag->length>0)
                        {
                      echo $error_number.'<font color = "RED"><b>--->Error!!! The intentional
element</b></font> "'.$intentional.'" <font color = "RED"><b>is drawn outside of the actor
boundary!!!</b></font>', "<br />";
                      echo '<div id="wrap">';
                  echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                  echo '<div id="example_'.$error_number.'" class="more">';
                  echo '<p><center><img src="./GifSR05.gif" width = "500" height="800"></center></p>';
                  echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                        echo '</div>';
                  echo '</div>';
                           $error_number++;

                                              }


          }


             }
           }
        }

}
```

```
 echo '<br />';
 echo '<HR>';


//CHECK #6 refinement/Decomposition links Extension out of an actor boundary

foreach ($Allboundaries as $bound)
{

   if($bound->hasChildNodes())
          {
             $boundaryKids=$bound->childNodes;
             echo 'The number of children in this boundary is :'.$boundaryKids->length , "<br />";
             foreach ($boundaryKids as $ielement)
             {

          $ielementType = $ielement->getAttribute('type');
                $intentional= $ielement->getAttribute('name');
                $id=$ielement->getAttribute('id');

                if ($ielement->hasChildNodes())
                            {

                   $ielementsChildren=$ielement->childNodes;
                                    //get the node of the the current iemement's encompassing actor

                   $childrenByTag=$ielement->getElementsByTagName('ielementLink');
                   if ($childrenByTag->length>0)
                                    {

                      foreach($childrenByTag as $iLink)
                      {

                         $iref = $iLink->getAttribute('iref');
                                                  //get the node of the the current iemementLink's
encompassing actor
                         $ielementLinkGGP=$iLink->parentNode->parentNode->parentNode;
                         $idActor=$ielementLinkGGP->getAttribute('id');
                                          $ActorName=$ielementLinkGGP->getAttribute('name');
                   $Found=FALSE;

                         foreach($Allielements as $intentionalElement)
                                                  {
                      $ide= $intentionalElement->getAttribute('id');

                      $nameIE= $intentionalElement->getAttribute('name');

                            $grandparentIE=$intentionalElement->parentNode->parentNode;
                            $grandparentIEID=$grandparentIE->getAttribute('id');

                            $grandparentIEName=$grandparentIE->getAttribute('name');
```

```
                        if ($iref==$ide)
                                                {
                            $Found=True;
                            if ($grandparentIE->nodeName=='actor')
                                                    {
                        if ($idActor!=$grandparentIEID)
                                                        {
                            echo $error_number.'<font color = "RED"><b>--->Error!!! A decomposition link
is extended beyond the boundaries of the actors!!<b></font> "', "<br />";
                            echo '<div id="wrap">';
                            echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                            echo '<div id="example_'.$error_number.'" class="more">';
                            echo '<p><center><img src="./GifSR06.gif" width = "500"
height="800"></center></p>';
                            echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                                echo '</div>';
                            echo '</div>';
                                    $error_number++;
                        }
                            }

                        }
                }
                    }

            }


        }

      }

   }

}
```

//Check #13

//the ielementLinks iref refer to any ielement but the ones as sg and goal leaves Except for goal and SG as root,

```
foreach ($Allboundaries as $bound)
{
   if ($bound->hasChildNodes())
          {
```

```
$boundaryKids=$bound->childNodes;
        if($boundaryKids->length>0)
                {
                  $ielementsInBound=$bound->getElementsByTagName('ielement');
                        $ielementLinks=$bound->getElementsByTagName('ielementLink');
                        if($ielementsInBound->length>0)
                        {
                           foreach($ielementsInBound as $ielement)
                                 {
                                    $id=$ielement->getAttribute('id');
                                          $intentional=$ielement->getAttribute('name');
                                          $type=$ielement->getAttribute('type');
                                          if ($type=='goal' or $type=='softgoal')
                                          {
//if it is SG or G so go and check if its id appears in one ielementLink iref

                  if($ielementLinks->length>0)
                                 {
                                    $found=FALSE; //will be True if we find that $id=$iref
                                                        foreach($ielementLinks as $ilink)
                                          {
                                             $iref=$ilink->getAttribute('iref');



                           //which are not referenced in any $id nowhere besides $ielement then ERROR
                           //if not found (!found) code then
                                       if ($id == $iref)
                                             {
                                                $found=True;
                                                break;
                                       }
                                          }
                                                                //AFTER FINISHING THE LOOP, if no match
was found so we should print the error msg
                                  if (!$found)
                                             {
                                                      echo $id. ",     ". $type. ",   ".$intentional;
                                                      echo $error_number.'<font color="RED"><b>--->
ERROR!! Softgoals and Goals should be decomposed further!!<b></font>' ,"<br />";
                                                      echo '<div id="wrap">';
                                 echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                                 echo '<div id="example_'.$error_number.'" class="more">';
                                 echo '<p><center><img src="./GifSR13.gif" width = "500"
height="800"></center></p>';
                                 echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                                      echo '</div>';
                                 echo '</div>';
                                             $error_number++;
```

```
                                        }


                                    }

                                }
                            }

                        }
                    }

            }

}



}
echo '<TABLE border = 1 cellpadding = 20>';
echo '<TR>';

echo '<TD> <center><b><font size = 14 color="GREEN">List of SR diagram
defects</font></b></center></TD>';
echo '</TR>';
echo '<TR>';

echo '<TD valign=top>',display_SR();
echo '</TD>';
echo '</TR>';
echo '</TABLE>';

?>
```

CheckTask.php file source code

```
<?php
function CheckTask($ielement, $ielementType, $intentional, $id, $bound, $error_label, $error_number)
{
        //echo 'This internal element is a '. $ielementType. ' and it is labeled as "'. $intentional. '" ,"<br
/>";
        if ($ielement->hasChildNodes())
                                                {
                $ielementsChildren=$ielement->childNodes;


        //we did grab only ielementLink children of the ielement since DOM xml php considers space,
text and tags as childNodes.
```

```
        //$childrenByTag are the specific children that we want to get from the list of children other than space and maybe text
                        $childrenByTag=$ielement->getElementsByTagName('ielementLink');
                        if ($childrenByTag->length>0)
                                        {

                           foreach($childrenByTag as $iLink)
                           {
                              $iLinkType = $iLink->getAttribute('type');
                              $iref = $iLink->getAttribute('iref');
                                                                $ILvalue = $iLink->getAttribute('value');

                              if($iLinkType=='decomposition' and $ILvalue=='or' )
                              {
                                        $done=False;
                                        $found=False;
                                        for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                                {
                                                $id2=$ielement2->getAttribute('id');
                                                if($id2==$iref)
                                                                                        {
                                                   $found=True;
                                                }
                                                                                                else
                                                                                                {

$ielement2=$ielement2->nextSibling;
                                                                                                        if ($ielement2
== False or $ielement2==NULL)
                                                                                                        {
                                                                                                           $done = true;
                                                                                                }
                                                }
                                        }
                                        if ($found)
                                        {
                                                $ielement2Type = $ielement2->getAttribute('type');
                                                $ielement2name = $ielement2->getAttribute('name');
                                                                                                switch
($ielement2Type)
                                                                                                {
                                 case "task":

                                                                                                echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused (inappropriately used) and was connecting a MEANS (TASK) to an END (TASK)!!</b></Font>',"<br />";
                                                                echo '<Font COLOR = PURPLE> <b>---> Attention! A means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
```

```
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./METaskT.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                            break;

                            case "goal":
                                                                                    echo '<Font COLOR =
GREEN>---><b> Successful Goal Decomposition!!</b></font>',"<br />";

                            break;

                            case "softgoal":
                            echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end
link was misused (inappropriately used) and was connecting a MEANS (TASK) to an END
(softgoal)!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./METaskSG.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;

                                                                                    break;

                                case "resource":
                            echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end
link was misused (inappropriately used) and was connecting a MEANS (TASK) to an END
(resource)!!</b></Font>',"<br />";
```

```
                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
                        $error_label++;
                                echo '<div id="wrap">';
                           echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                           echo '<div id="example_'.$error_number.'" class="more">';
                           echo '<p><center><img src="./METaskResource.gif" width = "500"
height="800"></center></p>';
                           echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                           echo '</div>';
                           echo '</div>';

                           $error_number++;

                                                                        break;
                        default:
                        echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                        }


                                }
                         }
                         elseif ($iLinkType=='decomposition' and $ILvalue=='and' )
                                                {
                                                        $done=False;
                                $found=False;
                                for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                {
                                        $id2=$ielement2->getAttribute('id');
                                        if($id2==$iref)
                                                                {
                                                $found=True;
                                        }
                                                                else
                                                                {

$ielement2=$ielement2->nextSibling;
                                                                        if ($ielement2
== False or $ielement2==NULL)
                                                                        {
                                                                           $done = true;
                                                                }
                                        }
                                }
                                if ($found)
                                {
```

```
                                        $ielement2Type = $ielement2->getAttribute('type');
                                        $ielement2name = $ielement2->getAttribute('name');

                                                                                switch
($ielement2Type)
                                                                                {
                                case "task":
                                echo '<Font COLOR = GREEN>---><b> Successful task
decomposition!!</b></font>',"<br />";
                                break;

                                case "goal":
                                echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A failure of
goal decomposition!! A task decomposition was misused to refine a goal!!!</b></Font>',"<br />";

                                   $error_label++;
                                         echo '<div id="wrap">';
                                   echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                   echo '<div id="example_'.$error_number.'" class="more">';
                                   echo '<p><center><img src="./TDTaskGoal.gif" width = "500"
height="800"></center></p>';
                                   echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                   echo '</div>';
                                   echo '</div>';

                                   $error_number++;
                                break;

                                case "softgoal":
                                echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A task
decomposition was misused to refine a softgoal!!!</b></font>',"<br />";
                                                                                echo '<Font COLOR =
PURPLE> <b>---> Attention! A task decomposition link should be used to decompose a TASK into sub
elements (sub-task, sub-goal, resource and softgoal) that are necessary to its
accomplishment!!!</b></Font>',"<br />";
                                $error_label++;
                                         echo '<div id="wrap">';
                                   echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                   echo '<div id="example_'.$error_number.'" class="more">';
                                   echo '<p><center><img src="./TDTaskSG.gif" width = "500"
height="800"></center></p>';
                                   echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                   echo '</div>';
                                   echo '</div>';
```

```
                    $error_number++;
                                                        break;

                        case "resource":
                    echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A task decomposition was misused to refine a resource!!!</b></font>',"<br />";
                                                                    echo '<Font COLOR = PURPLE> <b>---> Attention! A task decomposition link should be used to decompose a TASK into sub elements (sub-task, sub-goal, resource and softgoal) that are necessary to its accomplishment!!!</b></Font>',"<br />";
                    $error_label++;
                            echo '<div id="wrap">';
                       echo '<p><a href="#" id="example_'.$error_number.'-show" class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction explanation...</a></p>';
                       echo '<div id="example_'.$error_number.'" class="more">';
                       echo '<p><center><img src="./TDTaskResource.gif" width = "500" height="800"></center></p>';
                       echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink" onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                       echo '</div>';
                       echo '</div>';

                       $error_number++;
                                                        break;
                    default:
                    echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and unconventional element type was used!!</b></font>',"<br />";
                                                            }

                }


                                            }

                                            else
                                            {
                                               if($iLinkType=='contribution' )
                                                    {
                                                       $done=False;
                    $found=False;
                    for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                            {
                            $id2=$ielement2->getAttribute('id');
                            if($id2==$iref)
                                                            {
                               $found=True;
                            }
                            else
                                                                {
```

```
$ielement2=$ielement2->nextSibling;
                                                                                    if ($ielement2
== False or $ielement2==NULL)
                                                                                    {
                                                                                        $done =
true;
                                                                                }
                                                    }
                                        }
                                        if ($found)
                                        {
                                                $ielement2Type = $ielement2->getAttribute('type');
                                                $ielement2name = $ielement2->getAttribute('name');
                                                                                        switch
($ielement2Type)
                                                                                    {
                                case "task":

                                                                                echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! contribution link was misused and was
depicting a task contributing to another TASK!!</b></Font>',"<br />";
                                                        echo'<Font COLOR = PURPLE> <b>---> Attention!
contribution link should be used to indicate that an element is contributing positively or negatively to
satisficing a softgoal!!!</b></Font>',"<br />";

                                    $error_label++;
                                            echo '<div id="wrap">';
                                    echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                    echo '<div id="example_'.$error_number.'" class="more">';
                                    echo '<p><center><img src="./ContributionTaskTask.gif" width = "500"
height="800"></center></p>';
                                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                    echo '</div>';
                                    echo '</div>';

                                    $error_number++;
                                break;

                                case "goal":
                                                                                echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! contribution link was misused and was
depicting a task contributing to a goal!!</b></Font>',"<br />";
                                                        echo'<Font COLOR = PURPLE> <b>---> Attention!
contribution link should be used to indicate that an element is contributing positively or negatively to
satisficing a softgoal!!!</b></Font>',"<br />";
                                                                                $error_label++;
                                            echo '<div id="wrap">';
```

```
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./ContributionTaskGoal.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                                                                                        break;
                            case "softgoal":
                            echo '<Font COLOR = GREEN>---><b> Successful softgoal
refinement!!</b></font>',"<br />";

                            break;

                                case "resource":
                            echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! contribution
link was misused and was depicting a task contributing to a resource!!</b></Font>',"<br />";
                                                    echo'<Font COLOR = PURPLE> <b>---> Attention!
contribution link should be used to indicate that an element is contributing positively or negatively to
satisficing a softgoal!!!</b></Font>',"<br />";
                                                                                        $error_label++;
                                    echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./ContributionTaskResource.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                                                                                        break;
                            default:
                            echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                                        }
                                                                                    }


                                                                                }

                                                                            }
```

```
                    }
                }

                }

return array ($error_label, $error_number);
}

?>
```

CheckGoal.php file source code

```
<?php
function CheckGoal($ielement, $ielementType, $intentional, $id, $bound, $error_label, $error_number)
{

                if ($ielement->hasChildNodes())
                                {
                   $ielementsChildren=$ielement->childNodes;

                   $childrenByTag=$ielement->getElementsByTagName('ielementLink');
                   if ($childrenByTag->length>0)
                                        {

                      foreach($childrenByTag as $iLink)
                      {
                         $iLinkType = $iLink->getAttribute('type');
                         $iref = $iLink->getAttribute('iref');
                                                        $ILvalue = $iLink->getAttribute('value');

                         if($iLinkType=='decomposition' and $ILvalue=='and' )
                                                        {
                                $done=False;
                                $found=False;
                                for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                {
                                   $id2=$ielement2->getAttribute('id');
                                   if($id2==$iref)
                                                                {
                                      $found=True;
                                   }
                                                                else
                                                                {

$ielement2=$ielement2->nextSibling;
                                                                if ($ielement2
== False or $ielement2==NULL)
                                                                {$done = true;
```

```
                                                                                }
                                                                        }
                                                        }
                                                        if ($found)
                                                        {
                                                                $ielement2Type = $ielement2->getAttribute('type');
                                                                $ielement2name = $ielement2->getAttribute('name');
                                                                                                switch
($ielement2Type)
                                                                                                {
                                                                                                   case "task":
                                                                                                        echo '<Font
COLOR = GREEN><b>---> Successful Task decomposition!!</b></font>',"<br />";


                                                        break;

                                                        case "goal":
                                                                                                        echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A task decomposition link was
misused!!</b></Font>',"<br />";
                                                                                                        echo '<Font
COLOR = RED> <b>---> ERROR!! A goal should not be refined to subgoals!!</b></Font>',"<br />";


                                                                                echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                                                        $error_label++;
                                                                echo '<div id="wrap">';
                                                        echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                                        echo '<div id="example_'.$error_number.'" class="more">';
                                                        echo '<p><center><img src="./TDGoalGoal.gif" width = "500"
height="800"></center></p>';
                                                        echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                                        echo '</div>';
                                                        echo '</div>';

                                                        $error_number++;
                                                                                                        break;
                                                        case "softgoal":
                                                                                                        echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A task decomposition link was
misused!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = BLUE> <b>---> ERROR!! A goal-softgoal
relationship should illustrate a goal contributing to a softgoal via a contribution link!!</b></Font>',"<br
/>";
```

```
                                                    echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                                    $error_label++;
                                        echo '<div id="wrap">';
                                    echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                    echo '<div id="example_'.$error_number.'" class="more">';
                                    echo '<p><center><img src="./TDGoalSoftGoal.gif" width = "500"
height="800"></center></p>';
                                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                    echo '</div>';
                                    echo '</div>';

                                    $error_number++;
                                                                                    break;

                                        case "resource":
                                    echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A task
decomposition link was misused!!</b></Font>',"<br />";
                                    echo '<Font COLOR = BLUE> <b>---> ERROR!! There is no direct
relationship between a goal and a resource!!</b></Font>',"<br />";

                                                    echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                                    $error_label++;
                                        echo '<div id="wrap">';
                                    echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                    echo '<div id="example_'.$error_number.'" class="more">';
                                    echo '<p><center><img src="./TDGoalSResource.gif" width = "500"
height="800"></center></p>';
                                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                    echo '</div>';
                                    echo '</div>';

                                    $error_number++;
                                                                                    break;
                                    default:
                                    echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                                }

                                        }
                                }
```

```
                                                    elseif($iLinkType=='decomposition' and
$ILvalue=='or' )
                                                    {
                                                    $done=False;
                                $found=False;
                                for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                {
                                        $id2=$ielement2->getAttribute('id');
                                        if($id2==$iref)
                                                                {
                                           $found=True;
                                        }
                                                                else
                                                                {

$ielement2=$ielement2->nextSibling;
                                                                        if ($ielement2
== False or $ielement2==NULL)
                                                                        {$done = true;
                                                                }
                                        }
                                }
                                if ($found)
                                {
                                        $ielement2Type = $ielement2->getAttribute('type');
                                        $ielement2name = $ielement2->getAttribute('name');
                                                                switch
($ielement2Type)
                                                                {
                                                                   case "task":

                        echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end
link was misused (inappropriately used) and was connecting a MEANS (goal) to an END
(task)!!</b></Font>',"<br />";
                                                echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
                        $error_label++;
                                echo '<div id="wrap">';
                        echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                        echo '<div id="example_'.$error_number.'" class="more">';
                        echo '<p><center><img src="./MEGoalTask.gif" width = "500"
height="800"></center></p>';
                        echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                        echo '</div>';
                        echo '</div>';
```

```
$error_number++;
break;

case "goal":
                                                                    echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused
(inappropriately used) and was connecting a MEANS (goal) to an END (goal)!!</b></Font>',"<br />";
                                                    echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
$error_label++;
        echo '<div id="wrap">';
echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
echo '<div id="example_'.$error_number.'" class="more">';
echo '<p><center><img src="./MEGoalGoal.gif" width = "500"
height="800"></center></p>';
echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
echo '</div>';
echo '</div>';

$error_number++;
break;
case "softgoal":
                                                                    echo
$error_number. '<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused
(inappropriately used) and was connecting a MEANS (goal) to an END (softgoal)!!</b></Font>',"<br />";
                                                    echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
$error_label++;
        echo '<div id="wrap">';
echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
echo '<div id="example_'.$error_number.'" class="more">';
echo '<p><center><img src="./MEGoalSoftgoal.gif" width = "500"
height="800"></center></p>';
echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
echo '</div>';
echo '</div>';

$error_number++;
break;

        case "resource":
```

```
                    echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end
link was misused (inappropriately used) and was connecting a MEANS (goal) to an END
(resource)!!</b></Font>',"<br />";
                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
                    $error_label++;
                        echo '<div id="wrap">';
                    echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                    echo '<div id="example_'.$error_number.'" class="more">';
                    echo '<p><center><img src="./MEGoalResource.gif" width = "500"
height="800"></center></p>';
                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                    echo '</div>';
                    echo '</div>';

                    $error_number++;
                    break;
                    default:
                    echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                    }


                        }

                }

                else
                                                {
                                                   if($iLinkType=='contribution' )
                                                        {
                                                      $done=False;
                    $found=False;
                    for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                    {
                        $id2=$ielement2->getAttribute('id');
                        if($id2==$iref)
                                                                {
                            $found=True;
                        }
                        else
                                                                        {

$ielement2=$ielement2->nextSibling;
                                                                            if ($ielement2
== False or $ielement2==NULL)
                                                                            {
```

```
                                                                                $done
= true;
                                                                                        }
                                                        }
                                        }
                                        if ($found)
                                        {
                                                $ielement2Type = $ielement2->getAttribute('type');
                                                $ielement2name = $ielement2->getAttribute('name');

                                                                                        switch
($ielement2Type)
                                                                                {
                                                                                   case "task":

                                echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A
contribution link was misused and was connecting a  goal to a task!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./ContributionGoalTask.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                                break;

                                case "goal":
                                                                                        echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! There is no way to connect a goal to another
goal by any refinement link (There should not be any link connecting two goals)!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./ContributionGoalGoal.gif" width = "500"
height="800"></center></p>';
```

```
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                                break;
                                case "softgoal":
                                                                                                echo '<Font
COLOR = Green> <b>---> Successful contribution refinement!!</b></Font>',"<br />";

                                break;

                                        case "resource":
                                echo  $error_number.'<Font COLOR = RED> <b>---> ERROR!! A
contribution link was misused and was connecting a goal to a resource!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./ContributionGoalResource.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

        $error_number++;
                                break;
                                default:
                                echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                                        }

                                                                                }


                                                                        }

                                                        }

                                }


                        }
                }
```

```
return array ($error_label, $error_number);
}
?>
```

CheckSoftgoal.php file source code

```
<?php
function CheckSG($ielement, $ielementType, $intentional, $id, $bound, $error_label, $error_number)
{


                    if ($ielement->hasChildNodes())
                                        {
                       $ielementsChildren=$ielement->childNodes;

                       $childrenByTag=$ielement->getElementsByTagName('ielementLink');
                       if ($childrenByTag->length>0)
                                            {
                          foreach($childrenByTag as $iLink)
                          {
                             $iLinkType = $iLink->getAttribute('type');
                             $iref = $iLink->getAttribute('iref');
                                                                $ILvalue = $iLink->getAttribute('value');

                             if($iLinkType=='decomposition' and $ILvalue=='and' )
                                                                {
                                    $done=False;
                                    $found=False;
                                    for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                    {
                                       $id2=$ielement2->getAttribute('id');
                                       if($id2==$iref)
                                                                        {
                                          $found=True;
                                       }
                                                                        else
                                                                        {

$ielement2=$ielement2->nextSibling;
                                                                                if ($ielement2
== False or $ielement2==NULL)
                                                                                {
                                                                                   $done = true;
                                                                                }
                                       }
                                    }
                                    if ($found)
                                    {
```

```
$ielement2Type = $ielement2->getAttribute('type');
$ielement2name = $ielement2->getAttribute('name');
switch ($ielement2Type)
                                                                {
                                                                   case "task":

                        echo '<Font COLOR = GREEN><b>---> Successful Task
decomposition!!</b></font>',"<br />";


                        break;

                        case "goal":

                        echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A task
decomposition link was misused!!</b></Font>',"<br />";

                                                        echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                                        $error_label++;
                                echo '<div id="wrap">';
                        echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                        echo '<div id="example_'.$error_number.'" class="more">';
                        echo '<p><center><img src="./TDSoftgoalGoal.gif" width = "500"
height="800"></center></p>';
                        echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                        echo '</div>';
                        echo '</div>';

                        $error_number++;
                                                                        break;

                        case "softgoal":
                                                                                echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A task decomposition link was
misused!!</b></Font>',"<br />";

                                                        echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                                        $error_label++;
                                echo '<div id="wrap">';
                        echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                        echo '<div id="example_'.$error_number.'" class="more">';
```

```
                        echo '<p><center><img src="./TDSoftgoalSoftgoal.gif" width = "500"
height="800"></center></p>';
                        echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                        echo '</div>';
                        echo '</div>';

                        $error_number++;
                                                                        break;

                            case "resource":
                        echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A task
decomposition link was misused!!</b></Font>',"<br />";

                                                    echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                        $error_label++;
                            echo '<div id="wrap">';
                        echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                        echo '<div id="example_'.$error_number.'" class="more">';
                        echo '<p><center><img src="./TDSoftgoalResource.gif" width = "500"
height="800"></center></p>';
                        echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                        echo '</div>';
                        echo '</div>';

                        $error_number++;
                                                                        break;
                        default:
                        echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                    }
                            }
                    }


                                                        elseif($iLinkType=='decomposition' and
$ILvalue=='or' )
                                                        {
                                                            $done=False;
                        $found=False;
                        for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                {
                            $id2=$ielement2->getAttribute('id');
                            if($id2==$iref)
                                                                {
                                $found=True;
```

```
                                        }
                                        else
                                                                        {

$ielement2=$ielement2->nextSibling;
                                                                            if ($ielement2 ==
False or $ielement2==NULL)
                                                                                {
                                                                                    $done
= true;
                                                                            }
                                        }
                                }
                                if ($found)
                                {
                                        $ielement2Type = $ielement2->getAttribute('type');
                                        $ielement2name = $ielement2->getAttribute('name');
                                                                    switch ($ielement2Type)
                                                                        {
                                                                          case "task":

                                echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end
link was misused (inappropriately used) and was connecting a MEANS (softgoal) to an END
(task)!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
                                $error_label++;
                                    echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./MESGTask.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                                break;

                                case "goal":
                                                                                    echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused
(inappropriately used) and was connecting a MEANS (softgoal) to an END (goal)!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
                                $error_label++;
```

```
                                    echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./MESGGoal.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                                break;
                                case "softgoal":
                                                                                        echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused
(inappropriately used) and was connecting a MEANS (softgoal) to an END (softgoal)!!</b></Font>',"<br
/>";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";
                                $error_label++;
                                    echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./MESGSG.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;
                                break;

                                    case "resource":
                                echo '<Font COLOR = RED> <b>---> ERROR!! A Means-end link was
misused (inappropriately used) and was connecting a MEANS (softgoal) to an END
(resource)!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal relationship)!!!</b></Font>',"<br />";

                                break;
                                default:
                                echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                                    }
```

```
                                                                    }

                                elseif($iLinkType=='contribution' )
                                                                    {
                                                                                $done=False;
                                        $found=False;
                                        for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                                {
                                                $id2=$ielement2->getAttribute('id');
                                                if($id2==$iref)
                                                                                {
                                                   $found=True;
                                                }
                                                else
                                                                                {

$ielement2=$ielement2->nextSibling;
                                                                                if ($ielement2
== False or $ielement2==NULL)
                                                                                {
                                                                                        $done =
true;
                                                                                }
                                                }
                                        }
                                           if ($found)
                                           {
                                                $ielement2Type = $ielement2->getAttribute('type');
                                                $ielement2name = $ielement2->getAttribute('name');
                                                                                switch
($ielement2Type)
                                                                                {
                                                                                case "task":
                                                                                echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A contribution link was misused and was
connecting a  softgoal to a task as the softgoal is contributing to the task!!</b></Font>',"<br />";
                                                                    echo '<Font COLOR = PURPLE> <b>---> Attention! A
contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./ContributionSGTask.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';
```

```
                                $error_number++;

                                break;

                                case "goal":
                                                                                                echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A contribution link was misused and was
connecting a  softgoal to a goal as the softgoal is contributing to the task!!</b></Font>',"<br />";
                                                                echo '<Font COLOR = PURPLE> <b>---> Attention! A
contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./ContributionSGGoal.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;

                                break;
                                case "softgoal":
                                                                                                echo '<Font
COLOR = GREEN> <b>---> Correct softgoal refinement!!</b></Font>',"<br />";

                                break;

                                        case "resource":
                                echo '<Font COLOR = RED> <b>---> ERROR!! A contribution link was
misused and was connecting a softgoal to a resource as the softgoal is contributing to the
resource!!</b></Font>',"<br />";
                                                                echo '<Font COLOR = PURPLE> <b>---> Attention! A
contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";

                                break;
                                default:
                                echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                                                }

                                                                                        }

                                                                                }
```

```
                                        }

                        }

                }
        }
```

```
return array ($error_label, $error_number);
}
?>
```

CheckResource.php file source code

```
<?php
function CheckResource($ielement, $ielementType, $intentional, $id, $bound, $error_label,
$error_number)
{
if ($ielementType=='resource')
        {

                if ($ielement->hasChildNodes())
                                {
                   $ielementsChildren=$ielement->childNodes;

                   $childrenByTag=$ielement->getElementsByTagName('ielementLink');
                   if ($childrenByTag->length>0)
                                        {
                      foreach($childrenByTag as $iLink)
                      {
                         $iLinkType = $iLink->getAttribute('type');
                         $iref = $iLink->getAttribute('iref');
                                                        $ILvalue = $iLink->getAttribute('value');

                         if($iLinkType=='decomposition' and $ILvalue=='and' )
                                                        {
                                $done=False;
                                $found=False;
                                for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                {
                                   $id2=$ielement2->getAttribute('id');
                                   if($id2==$iref)
                                                                {
                                      $found=True;
                                   }
                                                                        else
                                                                        {
```

```
$ielement2=$ielement2->nextSibling;
                                                                        if ($ielement2
== False or $ielement2==NULL)
                                                                        {
                                                                           $done = true;
                                                                    }
                                                }
                                    }
                                    if ($found)
                                    {
                                            $ielement2Type = $ielement2->getAttribute('type');
                                            $ielement2name = $ielement2->getAttribute('name');
                                            switch ($ielement2Type)
                                                                                {
                                                                                   case "task":

                                    echo '<Font COLOR = GREEN><b>---> Successful Task
decomposition!!</b></font>',"<br />";


                                    break;

                                    case "goal":

                                    echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A task
decomposition link was misused!!</b></Font>',"<br />";

                                                        echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                                $error_label++;
                                        echo '<div id="wrap">';
                                    echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                    echo '<div id="example_'.$error_number.'" class="more">';
                                    echo '<p><center><img src="./TDResourceGoal.gif" width = "500"
height="800"></center></p>';
                                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                    echo '</div>';
                                    echo '</div>';

                                    $error_number++;
                                                                                break;

                                    case "softgoal":
                                                                                echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A task decomposition link was misused
(incorrectly used)!!</b></Font>',"<br />";
```

```
                                                    echo'<Font COLOR = PURPLE> <b>---> Attention! A
decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                                                                        $error_label++;
                                        echo '<div id="wrap">';
                                    echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                    echo '<div id="example_'.$error_number.'" class="more">';
                                    echo '<p><center><img src="./TDResourceSoftgoal.gif" width = "500"
height="800"></center></p>';
                                    echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                    echo '</div>';
                                    echo '</div>';

                                    $error_number++;



                                                                                        break;

                                        case "resource":
                                    echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A task
decomposition link was misused!!</b></Font>',"<br />";

                                                    echo'<Font COLOR = PURPLE> <b>---> Attention!
Decomposition link should be used to depict a relationship between a decomposed TASK and its
components (subtask, subgoal, resource and softgoal)!!!</b></Font>',"<br />";
                                    $error_label++;

            $error_number++;

                                                                                        break;
                                    default:
                                    echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                                    }
                                        }
                                }
                                                                elseif($iLinkType=='decomposition' and
$ILvalue=='or' )
                                                                {

                                        $done=False;
                                        $found=False;
                                        for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                                    {
                                            $id2=$ielement2->getAttribute('id');
                                            if($id2==$iref)
```

```
                                                                {
                                                    $found=True;
                                                }
                                                                        else
                                                                        {

$ielement2=$ielement2->nextSibling;
                                                                                if ($ielement2
== False or $ielement2==NULL)
                                                                                {
                                                                                    $done = true;
                                                                            }
                                                }
                                    }
                                    if ($found)
                                    {
                                            $ielement2Type = $ielement2->getAttribute('type');
                                            $ielement2name = $ielement2->getAttribute('name');
                                            switch ($ielement2Type)
                                                                        {
                                                                            case "task":

                                echo $error_number. '<Font COLOR = RED> <b>---> ERROR!! A Means-
end link was misused (inappropriately used) and was connecting a MEANS (resource) to an END
(task)!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A
means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as
means to achieve an end aka Goal)!!!</b></Font>',"<br />";
                                $error_label++;
                                        echo '<div id="wrap">';
                                echo '<p><a href="#" id="example_'.$error_number.'-show"
class="showLink" onclick="showHide(\'example_'.$error_number.'\');return false;">See correction
explanation...</a></p>';
                                echo '<div id="example_'.$error_number.'" class="more">';
                                echo '<p><center><img src="./MERTask.gif" width = "500"
height="800"></center></p>';
                                echo '<p><a href="#" id="example_'.$error_number.'-hide" class="hideLink"
onclick="showHide(\'example_'.$error_number.'\');return false;">Hide this explanation.</a></p>';
                                echo '</div>';
                                echo '</div>';

                                $error_number++;

                                break;

                                case "goal":
                                                                                    echo
$error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused
(inappropriately used) and was connecting a MEANS (resource) to an END (goal)!!</b></Font>',"<br />";
```

```
                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as means to achieve an end aka Goal)!!!</b></Font>',"<br />";
                                        $error_label++;

                $error_number++;
                                        break;
                                        case "softgoal":
                                                                                        echo $error_number. '<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused (inappropriately used) and was connecting a MEANS (resource) to an END (softgoal)!!</b></Font>',"<br />";
                                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as means to achieve an end aka Goal )!!!</b></Font>',"<br />";
                                        $error_label++;

                $error_number++;
                                        break;

                                                case "resource":
                                        echo $error_number.'<Font COLOR = RED> <b>---> ERROR!! A Means-end link was misused (inappropriately used) and was connecting a MEANS (resource) to an END (resource)!!</b></Font>',"<br />";
                                                                        echo '<Font COLOR = PURPLE> <b>---> Attention! A means-End link should be used to indicate the alternatives(TASKs) to achieve a GOAL (namely a task as means to achieve an end aka Goal)!!!</b></Font>',"<br />";
                                        $error_label++;

                $error_number++;
                                        break;
                                        default:
                                        echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and unconventional element type was used!!</b></font>',"<br />";
                                                                                                        }
                                                                                                }


                                                                        }
                                else
                                                                        {
                                                                           if($iLinkType=='contribution' )
                                                                                {
                                                                              $done=False;
                                        $found=False;
                                        for($ielement2=$bound->firstChild ; !$done&&!$found; )
                                                                                                {
                                                $id2=$ielement2->getAttribute('id');
                                                if($id2==$iref)
                                                                                                        {
                                                        $found=True;
```

```
                                        }
                                                                else
                                                                {

$ielement2=$ielement2->nextSibling;
                                                                        if ($ielement2
== False or $ielement2==NULL)
                                                                        {
                                                                           $done =
true;
                                                                }
                                        }
                                }
                                if ($found)
                                {
                                        $ielement2Type = $ielement2->getAttribute('type');
                                        $ielement2name = $ielement2->getAttribute('name');
                                        switch ($ielement2Type)
                                                                {
                                                                    case "task":
                                                                    echo '<Font
COLOR = RED> <b>---> ERROR!! A contribution link was misused and was connecting a  resource to a
task!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention!
A contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";

                                        break;

                                        case "goal":
                                                                        echo '<Font
COLOR = RED> <b>---> ERROR!! A contribution link was misused and was connecting a  resource to a
goal!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention!
A contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";

                                        break;
                                        case "softgoal":
                                                                        echo '<Font
COLOR = RED> <b>---> Incorrect softgoal refinement! A softgoal cannot be refined using a resource
element!!</b></Font>',"<br />";
                                                        echo '<Font COLOR = PURPLE> <b>---> Attention!
A softgoal can only be refined by using goal, softgoal, task and belief elements!!!</b></Font>',"<br />";
                                                                                echo
'<Font COLOR = PURPLE> <b>---> Attention! The permitted elements to contribute to a softgoal
satisficing are: goal, softgoal, task and belief!!!</b></Font>',"<br />";
                                        break;

                                                case "resource":
```

```
                                        echo '<Font COLOR = RED> <b>---> ERROR!! A contribution link was
misused and was connecting a resource to a resource!!</b></Font>',"<br />";
                                                                 echo '<Font COLOR = PURPLE> <b>---> Attention!
A contribution link should be used to refine a softgoal and to indicate the contributing elements to its
satisficing!!!</b></Font>',"<br />";

                                        break;
                                        default:
                                        echo '<Font COLOR = RED><b>---> ERROR!! An unexpected and
unconventional element type was used!!</b></font>',"<br />";
                                                                                                   }
                                                                                          }

                                                    }




                                                       }

                                   }


                          }
                       }
                                                     }

return array ($error_label, $error_number);
}
?>
```

Appendix C. Source code and screenshots of the GENERATi*ON tool

Appendix C contains the source code and the screenshots for the generated table of contents from the input iStarML files so it is relative to the GENERAT*i**ON tool which aims at deriviving summaries from the submitted i* models in order to let the users evaluate and review the content of their models for a subsequent model information validation. We continued the usage of PHP as the main programming language and once again JavaScript was used to define the showing and hiding (showHide) function for managing the appearance of certain content once a hypertext link is clicked on. CSS was used to describe how HTML elements are displayed on screen. Each uploaded iStarML file to this application will be stored in a folder in server called "UploadedFiles".

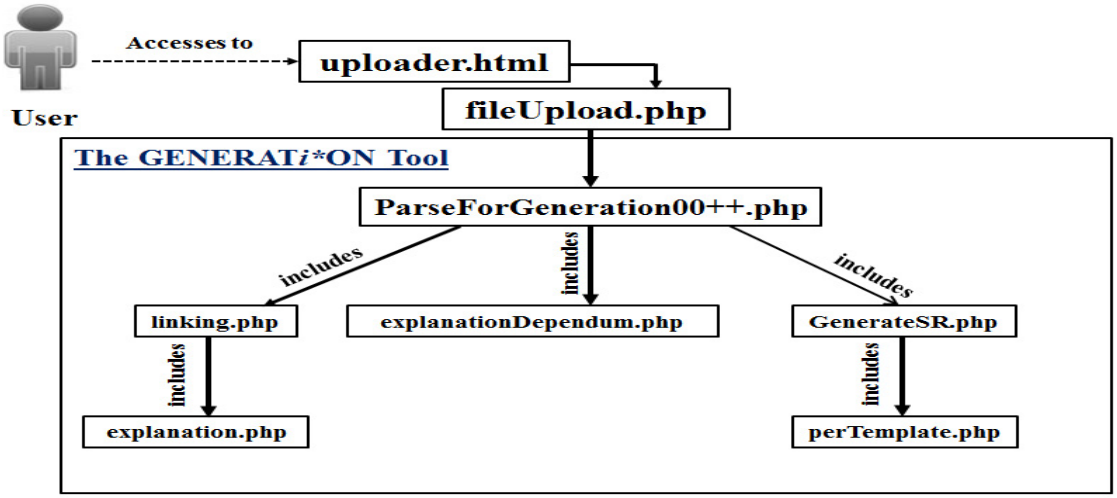


Figure C.1: Description of GENERAT*iON tool files and the interrelation (interaction) between them.**

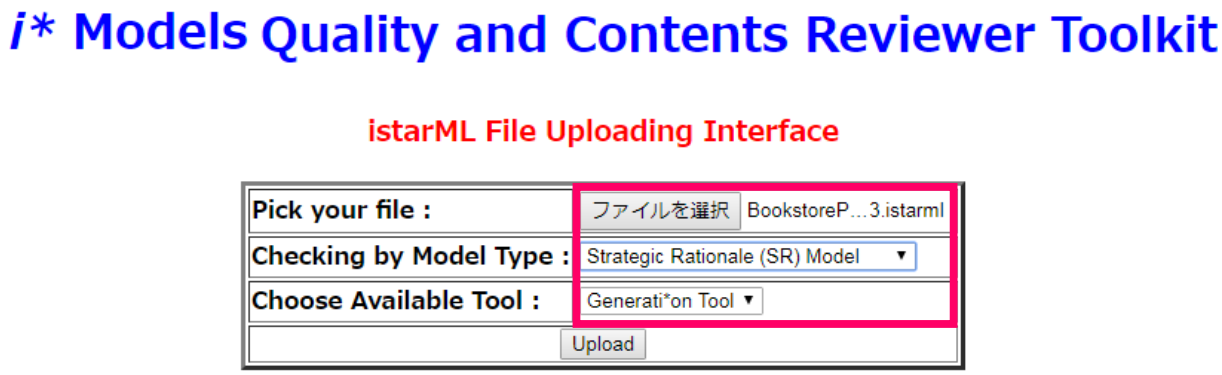


Figure C.2: An example of selecting an i* model "BookStoreParagraph3" to submit to the GENERAT*iON tool**

SUMMARY (Table Of Contents) of the i* Diagram from the file "BookStoreParagraph3.istarml"

I. ACTORS

II. DEPENDENCIES

Figure C.3: Generated Table of Contents where all information is encompassed either by actors or dependencies

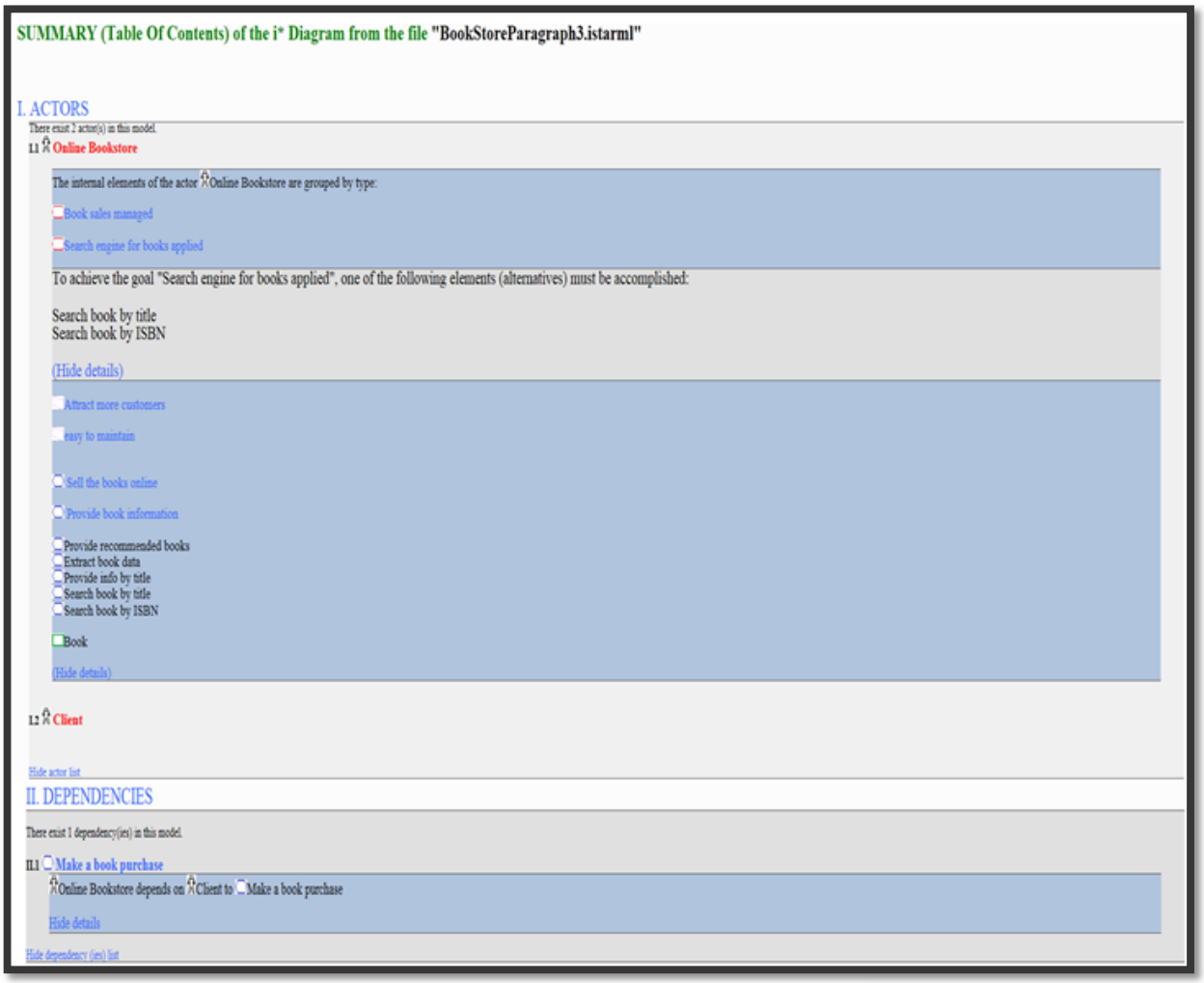


Figure C.4: Detailed view of the Table of Contents where all information about the internal elements of SR model as well as dependencies is presented to the user

GENERAT*i**ON tool files' source code

ParseForGeneration00++.php file source code

```
<script language="javascript" type="text/javascript">

function show(elementId) {

 document.getElementById(elementId).style.display="block";

}
```

```
function Hide(elementId) {
 document.getElementById(elementId).style.display="none";

}
</script>
<style type="text/css">
  /* This CSS is just for presentational purposes. */
  body {
    font-size: 75%;
    background-color: #fcfcfc; }
  #wrap {
    font: 1.2em/1 Times New Roman, Helvetica, sans-serif;

    margin: 0 auto;
    padding: 0em;
    background-color: #e0e0e0; }

        #wrap1 {
    font: 1.2em/1 Times New Roman, Helvetica, sans-serif;
    width: 98%;
    margin: 0 auto;
    padding: 0em;
    background-color:  #f0f0f0;


  }
```

```
        #wrap2{
   font: 1.2em/1 Times New Roman, Helvetica, sans-serif;
   width: 96%;
   margin: 0 auto;
   padding: 0em;
   background-color:   #B0C4DE; }

        #wrap3 {
   font: 1.2em/1 Times New Roman, Helvetica, sans-serif;
   width: 95%;
   margin: 0 auto;
   padding: 0em;
   background-color:  #f0f0f0;



}



/* This CSS is used for the Show/Hide functionality. */
.more {
   display: none;
   border-top: 1px solid #666;
   border-bottom: 1px solid #666; }
```

```
  a.showLink, a.hideLink {
    text-decoration: none;
    color: #36f;

    background: transparent url(down.gif) no-repeat left; }
  a.hideLink {
    background: transparent url(up.gif) no-repeat left; }
  a.showLink:hover, a.hideLink:hover {
    border-bottom: 1px dotted #36f; }
</style>

<?php
include("linking.php");
include ("explanationDependum.php");
include ("GenerateSR.php");
$dom = new DOMDocument();
$dom->preserveWhiteSpace = FALSE;
$name= $_REQUEST['filename'];
$Type= $_REQUEST['checkType'];
$dom->load("UploadedFiles/$name");
$xpath = new DOMXpath($dom);
$imgPath="./ICONS/";
$Allactors = $xpath->query('//actor');
$Allielements = $xpath->query('//ielement');
$Alldependencies = $xpath->query('//dependency');
$Alldependees = $xpath->query('//dependee');
```

```
$Alldependers =$xpath->query('//depender');

$Alldiagrams = $xpath->query('//diagram');

foreach($Alldiagrams as $diagram) {

$diagramName = $diagram->getAttribute('name');


echo '<h1><font color="GREEN"> Table Of Contents of the <i>i* </i>Diagram from the file
</font><font color="BLACK">"'.$diagramName.'.istarml"</font></h1>'  ,"<br />";

echo "<br />";

echo "<br />";


}


// It was a code to remove the unwanted elements. WE DIDN'T touch the source file!!!

$GraphicNodes = $dom->getElementsByTagName('graphic');


if ($GraphicNodes->length == 0)

{

        $done = True;  //the istarml File has no graphic nodes ()tags)

}

else

{

        $done = False;

}

   for (; !$done; )

   {
```

```
        $gnode=$GraphicNodes->item(0);



        $gnode->parentNode->removeChild($gnode);



        if ($GraphicNodes->length == 0)
        {
                        $done = True;  // no more graphic nodes left
                }



    }
$nbactors=1;
$listActors=0;
$link=1;
$SR=1;
echo '<font color = BLUE size="5" ><a href="#" id="example_'.$listActors.'-show" class="showLink"
onclick="show(\'example_'.$listActors.'\');return false;"> I. ACTORS  </a></font>';
echo '<div id="wrap1">';


       echo '<div id="example_'.$listActors.'" class="more">';
         echo 'There exist '.$Allactors->length. ' actor(s) in this model.  <br />';
                        foreach($Allactors as $actor) {


    $nameA = $actor->getAttribute('name');
    $idA = $actor->getAttribute('id');
```

```
        echo '<strong>I.'.$nbactors.'</strong> ';
        //echo '1'.$nbactors;
        echo "<img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >";
        if (!$actor->hasChildNodes()){
        echo '<font size = "4" color = RED><strong>'.$nameA.'</strong></font>';

        echo "<br />";}

 else

  {
            echo '<a href="#" id="example8_'.$link.'-show" class="showLink"
onclick="show(\'example8_'.$link.'\');return false;"> <font size = "4" color =
RED><strong>'.$nameA.'</strong></font></a>', "<br />";

            echo "<br />";
            echo '<div id="wrap2">';

           echo '<div id="example8_'.$SR.'" class="more">';

      GenerateSR($name, $idA, $nameA);
                  echo "<br />";
      echo '<a href="#" id="example8_'.$SR.'-hide" class="hideLink"
onclick="Hide(\'example8_'.$SR.'\');return false;">(Hide details)</a>';
     echo '</div>';
  echo '</div>';
  $SR++;
```

```
        echo"<br />";


    }
   $link++;
echo"<br />";
            $nbactors++;


}
         echo '<a href="#" id="example_'.$listActors.'-hide" class="hideLink"
onclick="Hide(\'example_'.$listActors.'\');return false;">Hide actor list</a>';
       echo '</div>';
    echo '</div>';


            echo "<br />";



// HERE STARTS THE DEPENDENCIES LIST:


echo"<br />";
$nbDep=1;
$linkD=1;
$listDep=0;
```

```
echo '<font color = BLUE size="5"><a href="#" id="example1_'.$listDep.'-show" class="showLink"
onclick="show(\'example1_'.$listDep.'\');return false;"> II. DEPENDENCIES  </a></font>';

echo '<div id="wrap">';

echo '<div id="example1_'.$listDep.'" class="more">';

echo "<br />";

//They are listed as below:

// pour afficher la liste des noms des dependencies une par une, suite a une icone dr dependum comme une
puce

echo 'There exist '.$Alldependencies->length. ' dependency(ies) in this model.  ',"<br />";  //list as follows

echo"<br />";

foreach ($Alldependencies as $dependency){

        $parentIE= $dependency->parentNode;


$nameD = $parentIE->getAttribute('name');

$TypeD = $parentIE->getAttribute('type');

echo '<strong>II.'.$nbDep.'</strong> ';



switch ($TypeD) {

   case "task":

      echo "<img src=\"$imgPath","tache.png\" width = \"19\" height=\"15\" > <font color= #00008B>";

      break;

   case "goal":

      echo "<img src=\"$imgPath","but.png\" width = \"19\" height=\"15\"> <font color= RED>";

      break;

   case "softgoal":

      echo "<img src=\"$imgPath","doux.png\" width = \"19\" height=\"15\"> <font color= PINK>";
```

```
        break;
            case "resource":
        echo "<img src=\"$imgPath","ressource.png\" width = \"19\" height=\"15\"><font color= GREEN>";
        break;
    default:
        echo " ";

}

echo '<a href="#" id="example7_'.$linkD.'-show" class="showLink"
onclick="show(\'example7_'.$linkD.'\');return false;"> </font> <strong> <font size=
"4">'.$nameD.'</font></strong></a>';

echo '<div id="wrap2">';

echo '<div id="example7_'.$linkD.'" class="more">';


elaborate ($name, $idA, $nameA, $TypeD, $nameD) ;

echo "<br />";

echo '<a href="#" id="example7_'.$linkD.'-hide" class="hideLink"
onclick="Hide(\'example7_'.$linkD.'\');return false;">Hide details </a>';

      echo '</div>';

   echo '</div>';

            $linkD++;

echo"<br />";

$linkD++;

$nbDep++;

}

echo '<a href="#" id="example1_'.$listDep.'-hide" class="hideLink"
onclick="Hide(\'example1_'.$listDep.'\');return false;">Hide dependency (ies) list</a>';
```

```
      echo '</div>';
   echo '</div>';


            echo"<br />";
            echo"<br />";
            echo"<br />";

?>
```

linking.php file source code

```
<?php
include("explanation.php");
 function Role ($filename,$id,$nameActor){
$dom = new DOMDocument();
$dom->preserveWhiteSpace = FALSE;
$dom->load("UploadedFiles/$filename");
$xpath = new DOMXpath($dom);
$imgPath="./ICONS/";
$actors = $xpath->query('//actor');
$ielements = $xpath->query('//ielement');
$Alldependencies = $xpath->query('//dependency');
$Alldependees = $xpath->query('//dependee');
$Alldependers =$xpath->query('//depender');

//STEP 1 We got the id of the actor of concern
```

```
//STEP 2 is looping through the list of dependers and find the dependers with aref equals to $id

echo '<font color = "#556B2F"><ins>'.$nameActor.'</ins> </font>--> "as Depender": ',"<br />";
echo "<br />";
$numdep=1;
$num_ders_cases=0;
foreach ($Alldependers as $depender){
$arefer=$depender->getAttribute('aref');

if ($arefer == $id) {

        //we checked the ielement of the current depender to get the needed info
        $grandparent=$depender->parentNode->parentNode;
        $gpname= $grandparent->getAttribute('name');
        $gpType= $grandparent->getAttribute('type');
        //go get the dependee aref
        $parent=$depender->parentNode;

        $DependeeList=$parent->getElementsByTagName('dependee');
        if($DependeeList->length==1){
        $NBee=$DependeeList->length;
   $Dependee=$DependeeList->item(0);
        $arefDependee=$Dependee->getAttribute('aref');

        foreach ($actors as $actor){
                $nameActeur = $actor->getAttribute('name');
```

```
        $idActeur = $actor->getAttribute('id');
                    if ($idActeur==$arefDependee){
                        $nomDependee=$nameActeur;
                        $num_ders_cases++;
                    }
            }

            echo $numdep.". <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >";
            Explain ($filename, $id, $nomDependee, $nameActor, $gpname, 1, $gpType);
            echo "<br />";
            $numdep++;



}



}



}
if ($num_ders_cases==0)

{echo "<font color = PURPLE> No instance of this actor plays the role of a depender in any
dependency!!</font>","<br />";
}

echo '<font color = "#556B2F"><ins>'.$nameActor.'</ins></font> --> "as Dependee": ',"<br />";
```

```
echo "<br />";
$numD=1;
$num_dees_cases=0;
foreach ($Alldependees as $dependee){
$arefee=$dependee->getAttribute('aref');
if ($arefee == $id) {

        $grandparent=$dependee->parentNode->parentNode;
        $gpname= $grandparent->getAttribute('name');
  $gpType= $grandparent->getAttribute('type');
        $parent=$dependee->parentNode;

        $DependerList=$parent->getElementsByTagName('depender');
        if($DependerList->length==1){
  $Depender=$DependerList->item(0);
        $arefDepender=$Depender->getAttribute('aref');
        //}


foreach ($actors as $actor){
                $nameActeur = $actor->getAttribute('name');
      $idActeur = $actor->getAttribute('id');
                if ($idActeur==$arefDepender){
                        $nomDepender=$nameActeur;
                        $num_dees_cases++;
```

```
                    }
            }

    echo '  ';

 echo $numD.". <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >";
 Explain ($filename, $id, $nomDepender, $nameActor, $gpname, 2 , $gpType);

            echo "<br />";
$numD++;
}

}

}
if ($num_dees_cases==0)

{echo "<font color = PURPLE> No instance of this actor plays the role of a dependee in any dependency!!</font>","<br />";
}
}

?>
```

explanation.php file source code

```
<?php
```

```
function Explain ($fname, $id, $otherActorName, $nameActor, $gpname, $numTemplate, $gpType)

{

$dom = new DOMDocument();

$dom->preserveWhiteSpace = FALSE;

$dom->load("UploadedFiles/$fname");

$xpath = new DOMXpath($dom);

$imgPath="./ICONS/";

$actors = $xpath->query('//actor');

$ielements = $xpath->query('//ielement');

$Alldependencies = $xpath->query('//dependency');

$Alldependees = $xpath->query('//dependee');

$Alldependers =$xpath->query('//depender');


//STEP 1 We got the id of the actor of concern

switch ($numTemplate) {


        case "1":

        // Before it was above but in the browser it displayed index problem of an empty arefDepender
and nDer are not defined


   $nameDependee= $otherActorName;


   switch ($gpType){

        case "task":

        echo $nameActor." depends on <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\"
>".$nameDependee. " to <img src=\"$imgPath","tache.png\" width = \"15\" height=\"20\"
>".$gpname,"<br />";

        break;
```

```
        case "resource":

        echo $nameActor." shall receive <img src=\"$imgPath","ressource.png\" width = \"15\"
height=\"20\" > ".$gpname." from  <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\"
>".$nameDependee,"<br />";

        break;

        case "goal":

    echo $nameActor."depends on  <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\"
>".$nameDependee. " to achieve the goal <img src=\"$imgPath","but.png\" width = \"15\" height=\"20\"
>".$gpname,"<br />";

    break;

        case "softgoal":

    echo $nameActor." depends on <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\"
>".$nameDependee. "to achieve the softgoal <img src=\"$imgPath","doux.png\" width = \"15\"
height=\"20\" >".$gpname,"<br />";

    break;



        default:

    echo " ";

        }

        break;



        //-------------------------------------------------------------------------------------------------------



        case "2":

        // Before it was above but in the browser it displayed index problem of an empty arefDependee
and nDee are not defined



    $nameDepender= $otherActorName;
```

```
 switch ($gpType){

case "task":

echo $nameActor. " shall (can/may) <img src=\"$imgPath","tache.png\" width = \"15\" height=\"20\" >".$gpname,"<br />";

break;

case "resource":

echo $nameActor." shall provide  <img src=\"$imgPath","ressource.png\" width = \"15\" height=\"20\" >".$gpname." to <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDepender,"<br />";

break;

case "goal":

echo $nameActor." shall attain the goal <img src=\"$imgPath","ressource.png\" width = \"15\" height=\"20\" >".$gpname,"<br />";

break;

case "softgoal":

echo $nameActor." shall be able to attain <img src=\"$imgPath","ressource.png\" width = \"15\" height=\"20\" >".$gpname,"<br />";

break;


default:

echo " ";

}

break;

//-------------------------------------------------------------------------------

default:

echo " ";

}
```

```
}
?>
```

explanationDependum.php file source code

```
<?php
function elaborate ($name, $idA, $nameA, $TypeD, $nameD)
{
$dom = new DOMDocument();
$dom->preserveWhiteSpace = FALSE;
$dom->load("UploadedFiles/$name");
$xpath = new DOMXpath($dom);
$imgPath="./ICONS/";
$actors = $xpath->query('//actor');
$ielements = $xpath->query('//ielement');
$Alldependencies = $xpath->query('//dependency');
//STEP 1 We got the id of the actor of concern

foreach ($Alldependencies as $dependency){
// check either i am looking at the very precise dependency specified by nameD and TypeD
$parent = $dependency->parentNode;
$DepName = $parent->getAttribute('name');
$DepType = $parent->getAttribute('type');
if($DepType==$TypeD and $DepName==$nameD){

        $dependerList= $dependency->getElementsByTagName('depender');
        $dependeeList= $dependency->getElementsByTagName('dependee');
```

```
        if ($dependerList->length ==1){
        $depender=$dependerList->item(0);
        $arefDepender= $depender->getAttribute('aref');
        }
        if ($dependeeList->length ==1){
        $dependee=$dependeeList->item(0);
        $arefDependee= $dependee->getAttribute('aref');
        }



// here we are looking for the actor whos id is matching the arefDepender
    $founder = FALSE;
        foreach($actors as $actor) {
    $nameA = $actor->getAttribute('name');
    $idA = $actor->getAttribute('id');
        if ($idA==$arefDepender){
                        $founder=True;
       $nameDepender=$nameA;

                break;
                }
        }

        // here we are looking for the actor whos id is matching the arefDependee
    $foundee = FALSE;
```

```
        foreach($actors as $actor) {
    $nameA = $actor->getAttribute('name');
    $idA = $actor->getAttribute('id');
        if ($idA==$arefDependee){
                        $foundee=True;
        $nameDependee=$nameA;

                break;
                }
        }

    switch ($TypeD){
        case "task":
        echo "<img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDepender." depends on <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDependee. " to <img src=\"$imgPath","tache.png\" width = \"19\" height=\"15\" />".$nameD,"<br />";
        break;
        case "resource":
        echo "<img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDepender." shall receive <img src=\"$imgPath","ressource.png\" width = \"19\" height=\"15\" >".$nameD. " from <img src=\"$imgPath","acteur.png\" width = \"19\" height=\"15\" />".$nameDependee,"<br />";

        break;
        case "goal":
    echo "<img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDepender." depends on <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDependee. " to achieve the goal  <img src=\"$imgPath","but.png\" width = \"19\" height=\"15\" /> ".$nameD,"<br />";

    break;
```

```
        case "softgoal":

        echo "<img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDepender." depends on <img src=\"$imgPath","acteur.png\" width = \"15\" height=\"20\" >".$nameDependee. " to satisfy the softgoal  <img src=\"$imgPath","doux.png\" width = \"19\" height=\"15\" /> ".$nameD,"<br />";

    break;


        default:

    echo " ";

        }


}

}


}

?>
```

GenerateSR.php file source code

```
<?php

include("perTemplate.php");

function GenerateSR($filen, $id, $nameActor) {

$dom = new DOMDocument();

$dom->preserveWhiteSpace = FALSE;

$dom->load("UploadedFiles/$filen");

$xpath = new DOMXpath($dom);

$imgPath="./ICONS/";

$actors = $xpath->query('//actor');

$ielementList = $xpath->query('//ielement');
```

```
$boundaryList = $xpath->query('//boundary');



//STEP 1 We got the id and the name of the actor of concern


$found=FALSE;
$nbactors=1;
foreach ($actors as $actor){
        $actorID= $actor->getAttribute('id');
        $actorName=$actor->getAttribute('name');
        if ($actorID == $id ){
                $found=True;


        echo "The internal elements of the actor  <img src=\"$imgPath","acteur.png\" width = \"15\"
height=\"20\" >".$nameActor. " are grouped by type: ","<br />";
   echo "<br />";


if ($actor->hasChildNodes()){
        $BoundariesList=$actor->getElementsByTagName('boundary');
        $boundary=$BoundariesList->item(0);
        $ielements=$boundary->childNodes;


        //preparing  arrays to collect the task name and initialize its index at 0
        $taskArray = array(array());
        $tai=0;


        $goalArray = array(array());
```

```
$gai=0;

$resourceArray = array(array());
$rai=0;

$sgArray = array(array());
$sai=0;



//looping through the list of ielements
foreach ($ielements as $ielement)
{

$ielementType = $ielement->getAttribute('type');
$intentional= $ielement->getAttribute('name');

$identifier=$ielement->getAttribute('id');



//we extracted the type and name of each ielement and we will group them by their type

switch ($ielementType){
case "task":

$taskArray[$tai][0]= $intentional;
```

```
//for taskArray, it has 4 rows: 1st row is for the name of the element, second row is for the hasChildNodes flag (0 -> no children, 1 -> has children),

//the 3rd row is root flag (0 ->not root node, 1 means it is a root node so it has no id) and the fourth row is for id of non root elements. if it is a root so in the fourth row and the concerned case we insert -1


//checking if task has children, if yes then $taskArray[$tai][1]=1; flag as 1

if ($ielement->hasChildNodes())

{

        $taskArray[$tai][1]=1;



}

else{

        $taskArray[$tai][1]=0;

        }


        if($identifier == ''){

           //$taskArray[$tai][2]=0;

                //$taskArray[$tai][3]=-1;

                $taskArray[$tai][2]=1;

                $taskArray[$tai][3]=-1;

        }

        else {$taskArray[$tai][2]=1;

             $taskArray[$tai][3]=$identifier;

        }
```

```
$tai++;



break;
case "resource":

$resourceArray[$rai][0]= $intentional;
//ACTUALLY resources are not decomposable
if ($ielement->hasChildNodes())
{
        $resourceArray[$rai][1]=1;

}
else{
        $resourceArray[$rai][1]=0;

        }
        if($identifier == ''){
                $resourceArray[$rai][2]=1;
                $resourceArray[$rai][3]=-1;

        }
        else {$resourceArray[$rai][2]=0;
        $resourceArray[$rai][3]=$identifier;}

$rai++;
```

```
break;
        case "goal":

        $goalArray[$gai][0]= $intentional;
        //checking if goal has children, if yes then $goalArray[$gai][1]=1; flag as 1
        if ($ielement->hasChildNodes())
        {
                $goalArray[$gai][1]=1;

        }
        else{
                $goalArray[$gai][1]=0;
                }
        if($identifier == ''){
                        $goalArray[$gai][2]=1;
                        $goalArray[$gai][3]=-1;

                }
                else {$goalArray[$gai][2]=0;
                $goalArray[$gai][3]=$identifier;}
        $gai++;

    break;
case "softgoal":
```

```
        $sgArray[$sai][0]= $intentional;
        //checking if sg has children, if yes then $sgArray[$tai][1]=1; flag as 1
        if ($ielement->hasChildNodes())
        {
                $sgArray[$sai][1]=1;

        }
        else{
                $sgArray[$sai][1]=0;
                }
                if($identifier == ''){
                        $sgArray[$sai][2]=1;
                        $sgArray[$sai][3]=-1;

                }
                else {$sgArray[$sai][2]=0;
                $sgArray[$sai][3]=$identifier;}


        $sai++;

break;

        default:
echo " ";
```

```
}

}

//printing all the existing elements in the goal array
echo '<strong> <font size= "3" color = "RED">I.' .$nbactors.'.1.GOALS </font></strong>', "<br />";
if ($gai>0){
$i=0;
$g=1;
$x=1;
$done=FALSE;

// while not done, check if $goalArray[$gai][1]=1 means the current element is having childNodes
while (!$done){

    $goal=$goalArray[$i][0];
    $goalID=$goalArray[$i][3];
    if($goalArray[$i][1]==1){
        if ($goalArray[$i][2]==1){

            echo "    ";

            echo "<img src=\"$imgPath","but.png\" width = \"19\" height=\"15\" >";
            echo '<a href="#" id="example10_'.$id.'_'.$g.'-show" class="showLink" onclick="show(\'example10_'.$id.'_'.$g.'\');return false;"> I.' .$nbactors.'.1.'.$x.' '.$goal.'</a>', "<br />";
```

```
            echo "<br />";
            echo '<div id="wrap3">';

           echo '<div id="example10_'.$id.'_'.$g.'" class="more">';
                Template($filen, $goalID, "goal", $goal, 1, 1);
                    echo "<br />";
                    echo '<a href="#" id="example10_'.$id.'_'.$g.'-hide" class="hideLink"
onclick="Hide(\'example10_'.$id.'_'.$g.'\');return false;"><font size = 3>(Hide details)</font></a>';
    echo '</div>';
  echo '</div>';
  $x++;

                echo "<br />";
                }

        else{echo "    ";
                    echo "<img src=\"$imgPath","but.png\" width = \"19\" height=\"15\" >";
                echo '<a href="#" id="example12_'.$id.'_'.$g.'-show" class="showLink"
onclick="show(\'example12_'.$id.'_'.$g.'\');return false;">  I.' .$nbactors.'.1.'.$x.' ' .$goal.'</a>', "<br />";
            echo "<br />";
            echo '<div id="wrap3">';

           echo '<div id="example12_'.$id.'_'.$g.'" class="more">';
                Template($filen, $goalID, "goal", $goal, 0, 1);
                    echo "<br />";
                    echo '<a href="#" id="example12_'.$id.'_'.$g.'-hide" class="hideLink"
onclick="Hide(\'example12_'.$id.'_'.$g.'\');return false;"><font size = 3>(Hide details)</font></a>';
    echo '</div>';
```

```
  echo '</div>';
                $x++;
                echo "<br />";
                    }
                }
                else {echo "    ";
                echo '<img src=\"$imgPath","but.png\" width = \"19\" height=\"15\" >
I.' .$nbactors.'.1.'.$x.' '.$goal,"<br />";
                $x++;
                }
                $i++;
                $g++;

                //$tai represents the number of tasks written to the array
                if ($i==$gai)
                {$done=True;}



        }
        }
        else {echo "    ";
        echo '<strong><font size= "2" color ="BROWN"><span>&#8594;</span>NOT
SPECIFIED!!</font></strong>',"<br />";}
        echo "<br />";
        echo '<strong> <font size= "3" color ="#dba2d1">I.' .$nbactors.'.2 SOFTGOALS
</font></strong>', "<br />";
        if ($sai>0){
        $i=0;
```

```
$s=1;
$done=FALSE;
// while not done, check if $goalArray[$tai][1]=1 means the current element is having childNodes
while (!$done){
$sg=$sgArray[$i][0];
$softID=$sgArray[$i][3];
if($sgArray[$i][1]==1){
if ($sgArray[$i][2]==1){
echo "    ";
echo "<img src=\"$imgPath","doux.png\" width = \"19\" height=\"15\" >";
echo '<a href="#" id="example14_'.$id.'_'.$s.'-show" class="showLink"
onclick="show(\'example14_'.$id.'_'.$s.'\');return false;"> I.' .$nbactors.'.2.'.$s.' ' .$sg.'</a>', "<br />";
echo "<br />";
echo '<div id="wrap3">';
echo '<div id="example14_'.$id.'_'.$s.'" class="more">';
Template($filen, $softID, "softgoal", $sg, 1, 2);
echo "<br />";
echo '<a href="#" id="example14_'.$id.'_'.$s.'-hide" class="hideLink"
onclick="Hide(\'example14_'.$id.'_'.$s.'\');return false;"><font size = 3>(Hide details)</font></a>';
echo '</div>';
echo '</div>';
$s++;
echo "<br />";
}
else {
```

```
                    echo "    ";

                    echo "<img src=\"$imgPath","doux.png\" width = \"19\" height=\"15\" >";

                     echo '<a href="#" id="example15_'.$id.'_'.$s.'-show" class="showLink"
onclick="show(\'example15_'.$id.'_'.$s.'\');return false;"> I.' .$nbactors.'.2.'.$s.' ' .$sg.'</a>', "<br />";

            echo "<br />";

            echo '<div id="wrap3">';


           echo '<div id="example15_'.$id.'_'.$s.'" class="more">';

                Template($filen, $softID, "softgoal", $sg, 0, 2);

                    echo "<br />";

                    echo '<a href="#" id="example15_'.$id.'_'.$s.'-hide" class="hideLink"
onclick="Hide(\'example15_'.$id.'_'.$s.'\');return false;"><font size = 3>(Hide details)</font></a>';

     echo '</div>';

  echo '</div>';

                    $s++;

                    echo "<br />";

                    }


                }

                else {echo "    ";

                echo '<img src=\"$imgPath","doux.png\" width = \"19\" height=\"15\"
>I.' .$nbactors.'.2.'.$s.' '.$sg,"<br />";

                $s++;}

                $i++;


                //$tai represents the number of tasks written to the array


                if ($i==$sai)
```

```
                {$done=True;}
        }
        }
        else{echo "    ";
        echo '<strong><font size= "2" color ="BROWN"><span>&#8594;</span>NOT SPECIFIED!!</font></strong>',"<br />";}

    echo "<br />";

        //we declare $i initialized at 0 and $done is false //$i will pcurs the array of tasks until it gets equal to $tai which counts the length of
        //the array (its number of elements)
        echo '<strong> <font size= "3" color = "BLUE">I.' .$nbactors.'.3 TASKS </font></strong>', "<br />";

        if ($tai>0){
        $i=0;
        $t=1;
        $done=FALSE;
        // while not done, check if $taskArray[$tai][1]=1 means the current element is having childNodes
        while (!$done){
                $task=$taskArray[$i][0];
                $taskID=$taskArray[$i][3];
                if($taskArray[$i][1]==1){
                        if ($taskArray[$i][2]==1){
                        echo "    ";
                        echo "<img src=\"$imgPath","tache.png\" width = \"19\" height=\"15\" > ";
```

```
        echo '<a href="#" id="example16_'.$id.'_'.$t.'-show" class="showLink"
onclick="show(\'example16_'.$id.'_'.$t.'\');return false;">I.' .$nbactors.'.3.'.$t.' ' .$task.'</a>', "<br />";

            echo "<br />";

            echo '<div id="wrap3">';


           echo '<div id="example16_'.$id.'_'.$t.'" class="more">';

                Template($filen, $taskID, "task", $task, 1, 3);

                    echo "<br />";

                    echo '<a href="#" id="example16_'.$id.'_'.$t.'-hide" class="hideLink"
onclick="Hide(\'example16_'.$id.'_'.$t.'\');return false;"><font size = 3>(Hide details)</font></a>';


    echo '</div>';

  echo '</div>';

  $t++;

  echo "<br />";


                }

                else {echo "    ";

                    echo "<img src=\"$imgPath","tache.png\" width = \"19\" height=\"15\" > ";


        echo '<a href="#" id="example17_'.$id.'_'.$t.'-show" class="showLink"
onclick="show(\'example17_'.$id.'_'.$t.'\');return false;"> I.' .$nbactors.'.3.'.$t.' ' .$task.'</a>', "<br />";

            echo "<br />";

            echo '<div id="wrap3">';


           echo '<div id="example17_'.$id.'_'.$t.'" class="more">';


                Template($filen, $taskID, "task", $task, 0, 3);
```

```
                    echo "<br />";

                    echo '<a href="#" id="example17_'.$id.'_'.$t.'-hide" class="hideLink"
onclick="Hide(\'example17_'.$id.'_'.$t.'\');return false;"><font size = 3>(Hide details)</font></a>';


    echo '</div>';

  echo '</div>';

                    $t++;

                    echo "<br />";

                     }

                }


                else {echo "    ";

                echo "<img src=\"$imgPath","tache.png\" width = \"19\" height=\"15\"
>I." .$nbactors.".3.".$t." " .$task,"<br />";

                $t++;}

                $i++;


                //$tai represents the number of tasks written to the array

                if ($i==$tai)

                {$done=True;}

        }


        }


        else{echo "    ";

        echo '<strong><font size= "2" color ="BROWN"><span>&#8594;</span>NOT
SPECIFIED!!</font></strong>',"<br />";}

        echo "<br />";
```

```
        echo '<strong> <font size= "3" color ="GREEN">I.' .$nbactors.'.4 RESOURCES
</font></strong>', "<br />";

    if ($rai>0){

    $i=0;

        $done=FALSE;

        while (!$done){

                $resource=$resourceArray[$i][0];

                if($resourceArray[$i][1]==1){

                        if ($resourceArray[$i][2]==1){

                        echo "    ";

                        echo '<img src=\"$imgPath","ressource.png\" width = \"19\" height=\"15\" ><a
href>I.' .$nbactors.'.4.'.' '.$resource."</a>","<br />";

                }

                }

                else {echo "    ";

                echo "<img src=\"$imgPath","ressource.png\" width = \"19\" height=\"15\"
>I." .$nbactors.".4."." " .$resource,"<br />";}

                $i++;

                //$rai represents the number of resources written to the array

                if ($i==$rai)

                {$done=True;}

        }

        }

        else{ echo "    ";

        echo '<strong><font size= "2" color ="BROWN"><span>&#8594;</span>NOT
SPECIFIED!!</font></strong>',"<br />";}

        }
```

```
}
$nbactors++;
}
}

?>
```

perTemplate.php file source code

```
<?php
function Template($filename, $idE, $typeIE, $nameIE, $root, $numTemplate){
$dom = new DOMDocument();
$dom->preserveWhiteSpace = FALSE;
$dom->load("UploadedFiles/$filename");
$xpath = new DOMXpath($dom);
$imgPath="./ICONS/";
$actors = $xpath->query('//actor');
$ielementList = $xpath->query('//ielement');
$AllIntentionalLinks = $xpath->query('//ielementLink');

//STEP 1 We got the id and the name of the actor of concern

$found=FALSE;
$label='';
foreach ($ielementList as $ielement){
        $type= $ielement->getAttribute('type');
```

```
        $label=$ielement->getAttribute('name');
        $idIE=$ielement->getAttribute('id');
        if ($root==0){
        if ($idIE == $idE ){
                // We have  found the non root element having the same id $found=True;

        $found=True;
        break;
        }
        }
        else{
                if ($type==$typeIE and $label==$nameIE){
                        //We have found the root element
  $found=True;
        break;

                }
        }
}
if (!$found) {
echo 'not found';
}
// maybe down

        $boundary=$ielement->parentNode;
        switch ($numTemplate) {
```

```
case "1":


// we should look in all the array created before or pass their content because they contain all the internal elements of the actor of our concern.

echo '<font size = 3> To achieve the goal "'.$nameIE. '", one of the following elements (alternatives) must be accomplished: ',"<br />";

 echo "<br />";


foreach($ielement->childNodes as $iLink) { //the $ielement here is the one that we found after the passage and comparison between idIE and idE so it is the ielement of concern

if ($iLink->nodeName == 'ielementLink')

{

$iLinkType = $iLink->getAttribute('type');

$iref = $iLink->getAttribute('iref');

if($iLinkType=='means-end')

{


$ielementList2=$boundary->childNodes;


foreach($ielementList2 as $iel2)

{

$id2=$iel2->getAttribute('id');

if($id2==$iref)

{


$found=True;

$name2=$iel2->getAttribute('name');
```

```
                        echo '-'. $name2 ,"<br />";
                        break;// stop and leave the loop
                        }
                }
        }

}
        }
        echo "</font>";
        break;
        case "2":

        // we should look in all the array created before or pass their content because they contain all the internal elements of the actor of our concern.
         echo '<font size = 3> The following element contribute to "'.$nameIE.'": ',"<br />";
   echo "<br />";
        $SGchildren=$ielement->childNodes;
        foreach($ielement->childNodes as $iLink) { //the $ielement here is the one that we found after the passage and comparison between idIE and idE so it is the ielement of concern

        if($iLink->nodeName == 'ielementLink')
        {
        $iLinkType = $iLink->getAttribute('type');
        $iLinkValue = $iLink->getAttribute('value');
        $iref = $iLink->getAttribute('iref');
        if($iLinkType=='contribution')
```

```
        {


                $ielementList2=$boundary->childNodes;
                foreach($ielementList2 as $iel2)
                {
                        $id2=$iel2->getAttribute('id');
                        if($id2==$iref)
                        {


                        $found=True;
                        $name2=$iel2->getAttribute('name');
                        echo '-'.$name2;
                        echo '  ';
                        echo "  ";
                        echo'<font color="GREEN">'.$iLinkValue.'</font>'  ,"<br />";
                        break;// stop and leave the loop
                        }
                }
        }

}
        }
        echo "</font>";
        break;
```

```
case "3":
$found = FALSE;

echo '<font size = 3>To perform "'.$nameIE. '" all the following elements must exist: ',"<br />";
echo "<br />";
// we should look in all the array created before or pass their content because they contain all the internal elements of the actor of our concern.

foreach($ielement->childNodes as $iLink) { //the $ielement here is the one that we found after the passage and comparison between idIE and idE so it is the ielement of concern
if ($iLink->nodeName == 'ielementLink')
{
$iLinkType = $iLink->getAttribute('type');
$iref = $iLink->getAttribute('iref');
if($iLinkType=='decomposition')
{
$done=False;
$found=False;

$ielementList2=$boundary->childNodes;
foreach($ielementList2 as $iel2)
{
$id2=$iel2->getAttribute('id');
```

```
                        if($id2==$iref)
                        {

                        $found=True;
                        $name2=$iel2->getAttribute('name');
                        echo '-'. $name2 ,"<br />";
                        break;// stop and leave the loop
                        }
                }
        }

    }
        }
        echo "</font>";
        default:
    echo " ";

    break;
        }
        if (!$found)
        {
        echo 'not found';
        }
}
?>
```