# Towards Anticipating Requirements Changes through Studies of the Future*

João Pimentel, Jaelson Castro, Hermano Perrelli, Emanuel Santos
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brazil
{jhcp, jbc, hermano, ebs}@cin.ufpe.br

Xavier Franch
Universitat Politècnica de Catalunya
Barcelona, Spain
franch@essi.upc.edu

*Abstract*— **Whilst it is considered a good practice to focus Requirements Engineering on current stakeholder needs, the high costs implied by requirements changes and the emergence of the Autonomic Computing paradigm raised the need for dealing with issues that are not currently requirements but that may come to be in the future. This work shows how foresight techniques can be used for requirements elicitation, and discusses the impacts of studying the future to that requirements engineering activity. In particular, it addresses the use of the Futures Wheel method to enrich goal models.**

***Keywords- Requirements elicitation, Requirements changes, Autonomic Computing, Self-adaptive systems, Studies of the future, Goal modeling.***

## I. INTRODUCTION

In the life cycle of a software product, maintenance is considered to be, usually, the most costly phase. This is largely due to the correction of errors that occurred on previous phases and to the increasingly dynamic context on which the systems run. The dynamic business and technological environment lead to the high occurrence of requirements changes. The changing of a requirement often leads to changes in other requirements, as well as in the system design, code and test cases. These changes are one of the main causes of software defects [11][18][37][38]. It is believed that the sooner a change is detected – on the product life cycle – the smaller is the cost of performing that change. Thus, if we can anticipate these changes on the initial development of the system, we can minimize their impact on the overall product life cycle.

In particular, there are some systems that are expected to analyze and implement some of these changes at runtime [2] – e.g., autonomic and self-adaptive systems. These systems are able to monitor the environment on which they are running, in order to identify the need of changing their behavior. To do so, it is required that these alternative behaviors are previously defined, at some level. Therefore, identifying the future changes of a system and defining how to handle these changes is a research challenge on information systems engineering.

In this paper we advocate the use of foresight methods during requirements elicitation in order to anticipate some of these changes. Although concerns about future requirements may already appear during normal requirements elicitation, the use of specific methods to capture possible future scenarios can provide a more detailed and realistic vision of the future.

Some works have already shown the benefits of using and adapting well-established methods from the social sciences. Based on those experiences, we believe that elaborating on the current methods of foresight used by social scientists and futurists is a better approach than creating entire new ones just for requirements elicitation. Thus, in this paper we summarize several foresight methods, providing an initial catalogue of foresight methods aligned to the needs of requirements elicitation. Moreover, we propose and analyze an approach that uses a specific method – Futures Wheel – and define how to use it to enrich a requirements model. We choose to express requirements with a goal-based language due to its suitability for expressing alternative behaviors. We exemplify the usage of this approach on the identification of changes for a television movies schedule system on a scenario of initial adoption of Digital Television (DTV).

The main contribution of this paper is to reduce the gap between requirements engineering and futures research. In particular, we present an approach to analyze a model of the future and adapt a requirements model to anticipate some of the changes that will occur during the system life cycle. Our objective is to reduce the occurrence of changes during the life cycle of a software product. In the context of self-adaptive systems, the usage of foresight methods described here may guide the definition of the adaptations to be performed.

This paper is structured as follows: Section 2 contextualizes the use of foresight methods in requirements elicitation and provides a summary of the suitable foresight methods. Section 3 discusses the general impact of considering the future during requirements elicitation. Section 4 describes with more details a specific foresight method and the goal model notation used to express requirements. In Section 5 we propose an approach that enriches a goal model of a system based on this foresight method. The usage of this approach is exemplified in Section 6. Section 7 discusses some related works. Section 8 concludes this paper and points out future work.

## II. DISCOVERING THE FUTURE

Discovering the current requirements of a system is already a complex task, but what to say about the future requirements? It is also a challenging task, especially considering that it is impossible to know for sure if a future event is really going to happen. On the other hand, the understanding of the future does not have to be as detailed as the understanding of the problem

* This work was partially supported by CAPES, CNPq, Erasmus Mundus External Cooperation Window - Lot 15 Brasil and the Spanish research project TIN2010-19130-c02-01

as it is nowadays. This is the case because the study of the future will be an additional source for requirements elicitation, rather than its basis.

**Definition 1 (Future event):** *a future event is an occurrence that is expected to take place in the future.*

According to [16], there are four dimensions to requirements elicitation, regarding problem analysis: Application domain, Problem to be solved, Business context and Stakeholder needs and constraints. If we want to elicit requirements dealing with future issues, we will need to consider these four dimensions in the future. A representation of the future becomes necessary. The future studies literature describes several techniques and methods [3][36] that allow the rational discovering of possible futures. These futures can be just one specific expected future or can be several different possible futures. They are often stated as diagrams, textual descriptions or mathematical representations. The foresight methods can be classified as qualitative or quantitative, and they may have other uses than just future studies, as is the case in Econometrics [21] and Scenarios [33], among others.

**Definition 2 (Representation of the future):** *a representation of the future is a model that describes a set of future events.*

A representation of the future can be either intentionally or accidentally created, and it can be of either a formal or an informal nature [8]. Hence, it may occupy any position on the axis of Figure 1. The best results would be obtained if a future model was formal and intentionally created, but not every project has sufficient resources to create such a model. Moreover, for some systems this may be particularly challenging. On these cases, the requirements engineer can collect some clues about the future while using normal elicitation techniques: listening to stakeholder comments during group sessions, reviewing the regulatory environment, analyzing the client plans, among others [10]. This model would be informal, and could be either accidentally or intentionally created.

Accuracy and precision are two main concepts in the studies of the future. Accuracy is a degree of how close the data is from the real data, while precision is a degree of how similar are different estimations of a data. Since requirements elicitation is concerned about the real world, the desired quality for future models used as input for elicitation is accuracy.

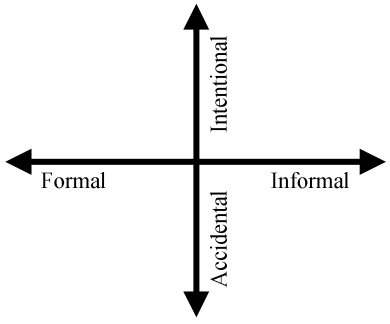


Figure 1 - Axis for characterization of a representation of the future

Gordon and Glenn [36] organizes foresight methods, or groups of methods, accordingly to what is intended to be achieved: (i) Collect judgments from experts; (ii) Forecast time series and other quantitative measures; (iii) Understand the linkages between events, trends and actions; (iv) Determine a course of action in the presence of uncertainty; (v) Portray alternative plausible futures; (vi) Reach an understanding of whether the future is improving; (vii) Track changes and assumptions and (viii) Determine the stability of a system. Four of these eight goals are strongly related to requirements elicitation: (i), (ii), (iii) and (v). Some usage examples are:

*(i) Collect judgments from experts* – can be used when it is necessary to reduce the uncertainty degree of a project;

*(ii) Forecast time series and other quantitative measures* – can be used to address scalability and security issues. For instance, by estimating the future load of a system;

*(iii) Understand the linkages between events, trends and actions* – can be used when it is needed to understand how the changing of a requirement will impact in other requirements;

*(v) Portray alternative plausible futures* – can be used when it is needed to understand a future usage scenario, enabling the anticipation of the changes required to support this scenario.

**Definition 3 (Foresight method):** *a foresight method is a means of creating a representation of the future.*

In this paper we are particularly interested in the category (v), since it has a more generic goal and can be applied on a

TABLE I. FUTURE METHODS CLASSIFIED ACCORDINGLY TO THEIR USE

| Category | Method |
|---|---|
| Collect judgments from Experts | Delphi<br>Futures Wheel<br>Participatory methods |
| Forecast time series and other quantitative measures | Econometrics forecast<br>Regression Analysis<br>Trend Impact Analysis<br>Structural Analysis |
| Understand the linkages between events, trends and actions | System Dynamics<br>Agent Modeling<br>Trend Impact Analysis<br>Cross Impact Analysis<br>Relevance Trees<br>Futures Wheel<br>Simulation Modeling<br>Multiple perspectives<br>Causal Layered Analysis<br>Field Anomaly Relaxation |
| Portray alternative plausible futures | Scenarios<br>Futures Wheel<br>Simulation and Gaming<br>Agent Modeling |

larger sample of systems. Nonetheless, Table 1 shows the methods of each one of these categories. Some of the methods fit into more than one category. In Table 1 the method *Interview* was excluded which appears in the original source, since it is actually a part of other methods, rather than a method by itself. Some slight changes on the names of the methods were also performed, to improve readability.

There are researches relating software engineering and some of these methods, like Delphi [5], System Dynamics [40], Agent Modeling [15] and Simulation Gaming [34]. Some of the foresight methods are even used for requirements elicitation, but not with the perspective of studying the future; e.g., Participatory methods and Scenarios.

The following subsections describe the 17 foresight methods presented in Table 1. This summary has been written with the purpose of being a catalogue on which a requirements engineer can identify the foresight method more suitable to the project being developed. These descriptions are mainly based on chapters of Glenn [19], which contains more detailed explanations and usage examples. The word *system* is used in this catalogue to mean the whole *social system* which the software system is going to be part of.

### A. *Delphi*

Category: Collect judgment from experts.

Description: Experts are asked to answer a questionnaire. These answers are consolidated and given back to the experts, which are then asked to reassess their opinion. This answer/feedback cycle is repeated until a consensus is raised or until there are no significant differences between cycles.

### B. *Futures Wheel*

Categories: Collect judgment from experts; Understand the linkages between events, trends and actions; Portray alternative plausible futures.

Description: This method consists of identifying and packing consequences of events or trends. The analyzed event, or trend, is defined and analyzed, leading to the definition of its consequences (or impacts).

### C. *Participatory methods*

Category: Collect judgment from experts.

Description: Participatory methods, in a future study context, are those that allow the participation of groups in order to explore possible and desirable futures. The groups can be located in the same place or not, and can range from small to large groups. Focus Groups, Opinion Polling, Charrette, Syncon, Public Delphi, Future Search Conference and Groupware are examples of participatory methods.

### D. *Econometrics forecast*

Category: Forecast time series and other quantitative measures.

Description: Econometrics groups methods that combine economic theory and statistics. Econometrics forecast is the use of these methods to forecast future developments in the economy. From economic models and past data one can discover changes in the patterns that emerge from the data.

### E. *Regression Analysis*

Category: Forecast time series and other quantitative measures.

Description: Regression Analysis is the use of historical data to generate equations that provide the value of a given variable at any time, depending on other variables. It is based on the mathematics of statistics.

### F. *Trend Impact Analysis*

Categories: Forecast time series and other quantitative measures; Understand the linkages between events, trends and actions.

Description: In Trend Impact Analysis the impact of unprecedented events is incorporated into a time-series (the trend). The first step is to identify the curve that best fits historical data and provide its extrapolation. The second and last step is to adapt the extrapolation curve, according to each unprecedented future event.

### G. *Structural Analysis*

Category: Forecast time series and other quantitative measures.

Description: In Structural Analysis, a system is described as matrixes that represent and link all its constitutive elements. The first step is to identify the system variables. The second step is to describe the relationship between the variables. Finally, the third step is to identify which are the essential variables.

### H. *System Dynamics*

Category: Understand the linkages between events, trends and actions.

Description: System Dynamics represents a system as stocks, flows and feedback loops. It produces Causal Links diagrams, Stock and Flow diagrams and a computer-generated simulation. With this simulation one can foresee how the system will perform, given some variables.

### I. *Agent Modeling*

Categories: Understand the linkages between events, trends and actions; Portray alternative plausible futures.

Description: In a forecast context, Agent Modeling is the use of computer-generated agents to simulate the behavior of a system. In this simulation, one can insert a future event and observe how the system will behave.

### J. *Cross Impact Analysis*

Category: Understand the linkages between events, trends and actions.

Description: In this method, a probability is associated to each event, from a set of future events. Also it is defined how the occurrence of each event impacts probabilities of the other events. Then a simulation is run to determine a new probability to each event, based on the frequency of their occurrence during the simulation.

*K. Relevance trees*

Category: Understand the linkages between events, trends and actions.

Description: A Relevance tree is a hierarchical structure where abstract concepts are refined into clear and, preferably, quantified terms. With this tree, one can have a clear understanding on which factors influence some future condition.

*L. Simulation Modeling*

Category: Understand the linkages between events, trends and actions.

Description: This method consists of mathematically describing the system being analyzed, using logic inferences instead of using just statistics (which is the case on Regression Analysis). With this system description one can simulate how the system will behave in the future.

*M. Multiple perspectives*

Category: Understand the linkages between events, trends and actions.

Description: Multiple perspectives is the use of three different and mutually supportive perspectives when trying to understand the impact of some future event onto a system. Usually, the event is the adoption of a new technology. These perspectives are, namely, Technical perspective, Organizational perspective and Personal perspective.

*N. Causal Layered Analysis*

Category: Understand the linkages between events, trends and actions.

Description: This method seeks to provide a deep understanding of the present and the past of a given subject. With this deep understanding, the researchers can then generate a vision of the future. The understanding is obtained by dividing and characterizing the studied subject in four layers: Litany – the most superficial one; Social system and structure; Worldview; Myth and metaphor.

*O. Scenarios*

Category: Portray alternative plausible futures.

Description: A scenario is a narrative description of what might unfold when an event occurs or a trend evolves, on which one can see the problems, challenges and opportunities regarding this future. Different authors have proposed methods for scenarios creation.

*P. Field Anomaly Relaxation*

Category: Understand the linkages between events, trends and actions.

Description: It is a method for projecting descriptions of evolution lines in a given system, which is called a field. This method comprises four steps: Create a view of future contexts in the field of concern; Construct a symbolic language to describe whole contextual patterns; Filter out non-coherent configurations and Compose scenarios.

*Q. Simulation and Gaming*

Category: Portray alternative plausible futures.

Description: Simulation and Gaming can experiment on different courses of action and, consequently, identify and analyze different alternatives.

## III. FUTURE-INFLUENCED REQUIREMENTS

The foresight methods described in the previous Section can be used to create a representation of the future. In this section we discuss how those representations can affect the requirements of a system.

Requirements changes on software that is already developed or in development may cause major problems in the development project, since these changes may provoke changes in its design, code, tests, and so on [18]. If we have a representation of the future, we can minimize these problems foreseeing some of the changes that will possibly be required, before the software development starts. The analysis of these foreseen changes can reflect in the requirements document on three possible ways: by provoking (i) the creation of new requirements; (ii) the exclusion of requirements that already exist and (iii) changes on requirements that already exist.

For the sake of analysis, these changes may be stored as a list of changes to be performed. Then, they may be analyzed and prioritized as conventional requirements. A key factor to be observed on this prioritization is the probability of the future event that provoked the change. If a change is derived from a future event with low probability, its priority is likely to be low as well. After the prioritization, the selected changes can be performed and the other changes can be stored in a backlog. Alternatively, the mechanisms for requirements variability [13] – usual on Software Product Lines – can be used to automatically perform these changes, enabling an easier analysis of the impact of these changes on the requirements document.

In the remainder of this section we are going to present some examples of future events and how they may affect the requirements of an enterprise information system. The following statement is an example of a foreseen change that possibly generates a new requirement:

"In 1 or 2 years from now our company will have a new department for sales, which currently is part of the marketing department."

Once a requirements engineer has this kind of information, she may create a new requirement to deal with it. The requirement may be very specific – like "Allow the creation of a sales department" – or more general – like "Allow the creation, changing and exclusion of departments".

The non-functional requirements (NFR) are usually more difficult to change on an already developed system, since they usually affect the system as a whole. Therefore, higher gains would be obtained by anticipating these changes. An example of future event related to NFR is "The expected load of the system on five years after deployment is of 500.000 transactions per day." Being aware of this data, the scalability NFR may be properly refined and later addressed with a

scalable architecture. Instead, the system could be developed on a non-scalable architecture and, when the load gets too high, the cost for scaling it would be too high as well.

There are occasions where the change has a strict deadline to happen. This is the case, for instance, when the software needs to obey government regulations or industrial standards. With a requirement explicitly stating that some change will occur in the future, e.g. "Starting in February 1st, 2012, the system will have to send the company's balance sheet to the regulatory agency in a monthly basis", the system can be designed to already support that functionality. On these cases, the system may (i) automatically adopt the new functionality when the given date comes; (ii) prompt for user confirmation or (iii) wait until an user or administrator explicitly require this functionality to be turned on.

One particular kind of future-related requirements are those that deal with the software deployment impact. Future models can be used to predict how a software system will change the environment on which it will be used, and requirements can be incorporated in order to minimize the bad impacts and maximize the good ones.

Most of the examples presented on this section may already take place during conventional requirements elicitation. However, this happens most often on an ad hoc basis, using informal and accidental future models. In this paper we intend to promote a systematic study of the future, moving towards formal and intentional models. This is expected to improve the precision of the model of the future and, thus, improve the quality of the requirements elicitation.

## IV. MODELING FUTURE AND REQUIREMENTS

In this section we present the Futures Wheel foresight method and its notation for writing models of the future. Then, we describe a goal modeling notation, which can be used to express system requirements. Both notations will be used in our approach, which is detailed in the next section.

We identified futures wheel as a suitable foresight method for requirements elicitation since (i) it provides a clear picture of the future events that may impact the system, (ii) it is easy to be understood and used by stakeholders and (iii) it requires less effort than the other approaches, therefore not compromising the project schedule.

### A. Futures Wheel

Futures wheel is a foresight method that provides a model of the future based on the consequences of an event or trend. It is a subjective and qualitative method, relying on the experience and knowledge of the participants. Its low complexity allows its usage without requiring a specialized training to be carried on. Nonetheless, it requires a deep understanding of the problem domain being analyzed, so that the generated future model may be as accurate as possible. Therefore, it is important the strong involvement of representatives of the client or domain experts during the model generation.

This method can be performed either by a single person – e.g., the requirements analyst of a project – or it can be performed collaboratively, usually in a meeting lead by a mediator. The method itself consists of two steps.

The first step is to identify trends or events that are likely to occur in a near future and that are related with the problem domain. A trend is something that has already started and is growing stronger, like "Use of electric car" or "Stream of live videos on the Internet". A future event is simply something that is expected to happen - e.g. "The entire population of Country X will have access to the Internet" or "A woman will be elected president of the USA". For the sake of simplicity, we will hereafter refer to trend or future event only as event.

The second step is to refine the event, adding its consequences. For each event, we will ask "what are the impacts, or consequences, of this event"? Then, for each consequence, identify the secondary consequences – i.e., the consequences of the consequences –, the tertiary consequences, and so on.

There will be one futures wheel model for each event - a graph in which one can see what the possible consequences of that event are. The event is represented by a circle with a thick border. The consequences are represented by a circle with a normal border. The main event is linked to the primary consequences by a single line arrow; the primary consequences are linked to the secondary consequences by a double line arrow, and so on. This notation is depicted in Figure 2. The circle with a thick border shows that *A* is the event being analyzed. The single line arrows indicate that *B* and *C* are the primary consequences of *A*. The double line arrows indicate that *X* is a consequence of *B* and of *C*, and that *Y* is a consequence of *C* – Therefore, X and Y are secondary consequences. Note that there is no way of representing that two or more consequences are alternative, mutually exclusive, or any other kind of relationship but that of consequence.

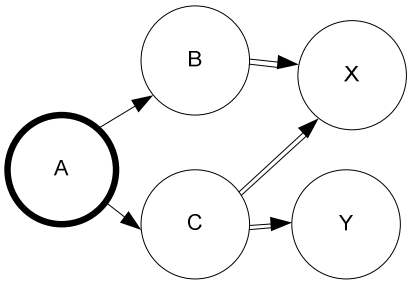


Figure 2 – Example of the futures wheel notation

### B. Goal modeling

In goal-oriented approaches [4], the role of Requirements Engineering (RE) is related to the discovery, the formulation, the analysis and the agreement of *what* is the problem being solved, *why* the problem must be solved and *who* is responsible for solving the problem. As the use of goals grew in RE, there are several techniques where goals are used as abstraction, including KAOS [1], NFR Framework [27], *i** [12], V-Graph [41] and Techne [17].

Among these approaches, we chose *i**, which will be briefly presented in this subsection. Besides being a widespread goal modeling language, *i** provides a suitable mechanism to represent alternative behaviors of a system,

through means-end links. This characteristic makes it easier to integrate the future-influenced requirements with the current goal model of the system.

*i** defines models to describe both the system and its environment in terms of intentional dependencies among strategic actors [12] (*who*). There are two different diagrams, or views, of an *i** model: the Strategic Dependency (SD) view presents only the actors and the dependency links amongst them, whilst the Strategic Rationale (SR) view shows the internal details of each actor. Within a SR diagram it is defined *why* each dependency exists and *what* is required to fulfill them.

Besides the actor, there are four key elements in *i**: goals, softgoals, tasks and resources. The goals represent the strategic interests of actors, that is, their intentions, needs or objectives to fulfill their roles within the environment in which they operate. Softgoals also represent the strategic interests of the actors, but in this case these interests are of subjective nature. They are not measured in concrete terms, but are generally used to describe the actors' desires related to quality attributes of their goals. The tasks represent a way to perform some activity, i.e., they show how to perform some action to obtain satisfaction of a goal or softgoal. The resources represent data or information that an actor may provide or receive.

There is one kind of dependency related to each one of these four elements. A goal dependency states that the depender needs the dependee to satisfy a goal for him. Similarly, in a softgoal dependency the depender needs the dependee to meet a softgoal. In a task dependency, the dependee is asked to perform an activity for the depender. A resource dependency express that the depender needs some resource that may be provided by the dependee.

In the SR diagram, the actor will be detailed using task-decomposition, means-end and contribution links (Figure 3). The *means-end* links defines which alternative tasks (means) may be performed in order to achieve a given goal (end) (e.g., *Task T1* is a possible means to achieve the goal *Goal G1*). The *task-decomposition* links describes what should be done to perform a certain task (e.g., *Task T1* is decomposed onto *Task T2* and *Task T3*). Finally, the contributions links suggest how a task can contribute (positively or negatively) to satisfy a softgoal (e.g., the task *Task T2* contributes negatively to the softgoal *Softgoal S1*). These contributions allow the selection of alternative tasks driven by the satisfaction of softgoals, which includes non-functional requirements. Lastly, the resource dependency between *Actor A1* and *Actor A2* means that, to perform *Task T3*, the actor *Actor A1* needs a resource *Resource R1* that can be provided by the execution of *Task T4* by *Actor A2*.

## V. INTEGRATING FUTURES WHEEL AND GOAL MODELS

Here we present our approach to analyze the futures wheel models and adapt the goal model to reflect the expected changes in the problem domain.

Our approach has two steps, assuming that a goal model of the system is available. The first step is to build an extended futures wheel model, and the second step is to alter the system goal model so that it can deal with the consequences expressed in the futures wheel model.

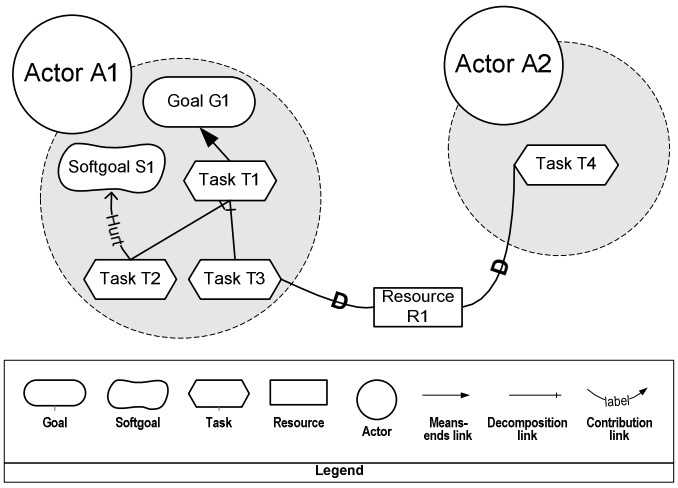


Figure 3 – Example of a goal model to illustrate its main concepts

### A. Build extended futures wheel models

We start as we were building a normal futures wheel model: identify the events and its consequences. At this point, there is still a large gap between the consequences and the system requirements. So, for each leaf consequence, i.e., the consequences that have no further consequences, we ask "how does this consequence affect the system"? We call these the direct consequences, since they are directly related to the system. To make explicit which are the direct consequences, we represent them as circles with a dashed border.

When identifying the consequences and the direct consequences, we should consider the four requirements elicitation dimensions presented in [16]: Application domain, Problem to be solved, Business context and Stakeholder needs and constraints.

Figure 4 shows an example of an extended futures wheel model. The consequences X, W and Z were, at first, leaf consequences. Then we added the direct consequences P, Q and S, which are consequences directly related to the system. Not necessarily all leaf consequences have direct consequences, as is the case of the consequence W.

The metamodel of this extended futures wheel notation is presented in Figure 5, using the Unified Metamodel Language (UML) [31]. Therefore, an extended future wheel model is an instance of this metamodel. The Event class is a singleton, since we are going to define only one future event for each model. An event can have an indefinite number of consequences, whilst each consequence is a consequence of a single event. Each consequence can also have an indefinite number of (sub-)consequences. On the other hand, each consequence can be a consequence of an indefinite number of (super-)consequences. Similarly, each consequence can have an indefinite number of direct consequences, and each direct consequence may be a consequence of an indefinite number of consequences. Each class of this metamodel has a String attribute to represent the description of their instances – for instance, to describe what is the future event being modeled.

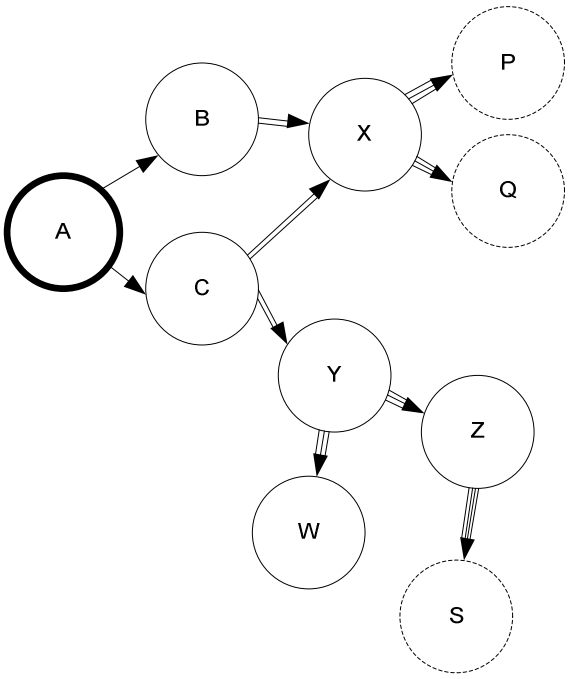


Figure 4 – Example of the extended futures wheel model notation

### B. *Adapt goal model*

For each direct consequence identified in the extended futures wheel model, we analyze how the system can be altered in order to deal with these consequences. The question to be answered is "How the system can address this direct consequence"?

The answers are expressed in the goal model, in which we may add, change or delete some elements and associations. This is a subjective activity, which needs to be performed on a case-by-case reasoning.

In order to preserve traceability information, one can use a traceability list, relating the direct consequences to the changes that were made in the goal model. This information would provide a rationale to some elements of the goal model.

Ideally, the system should be implemented so that it can deal with all of the foreseen changes. But in practice, there must be a compromise between the probability of the direct consequence to occur and the cost of implementing the system in a way that it can already deal with that consequence. If the probability is too low and the cost is too high, the risk of anticipating the change may be higher than the risk of not anticipating it.

## VI. CASE STUDY

In order to analyze the suitability of our approach, we developed a case study in the Brazilian Television (TV) movies domain, a system called "Movies For Me". In Brazil, the television is broadcast through ground antennas, which is called Terrestrial TV, and the population does not pay any fee to get access to the open channels. There is also subscribed (paid) TV, through cable, radio or satellites, but its reach is by far not as significant as the free TV.

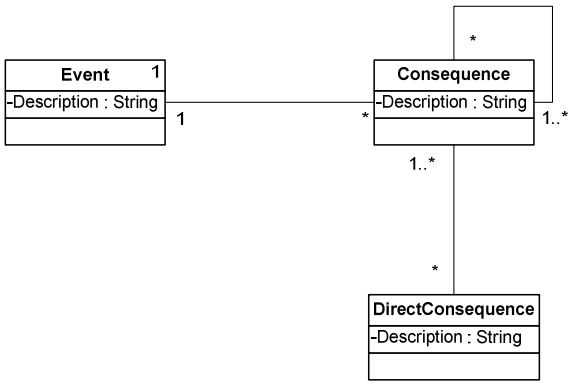


Figure 5 – Metamodel of the extended futures wheel notation

The Movies For Me system displays in a website the schedule of movies that will be showed in TV, with information like cast, director, plot and pictures. Its goal model is depicted in Figure 6.

The main goal of Movies For Me is "Inform the TV movies schedule", which can be achieved through the "Discover the TV movies schedule" and "Display the TV movies schedule" tasks. The task of discovering the schedule is achieved with the tasks "Discover the movies of Channel A" and "Discover the movies of Channel B". Each of them is decomposed in parsing the respective channel website, "Get movies description", "Get movies date and time" and "Get movies pictures". In Brazil, there are five major TV channels with countrywide reach and that show movies, but for the sake of simplicity we are presenting the goal model considering only two channels.

The "Display the TV movies schedule" goal is achieved with the "List movies on a website" task and the movies can be grouped by day of exhibition or by channel.

In 2008, Brazil started the adoption of Digital Terrestrial TV. Nowadays, this adoption is still restricted to a few estates, and in these estates there is a low share of watchers with a digital receiver. Even so, the successful adoption of Digital Terrestrial is expected to occur in a couple of years, when it will get countrywide reach and most of the television sets will come from factory already equipped with a digital receiver. So, we decided to consider the impacts that this event would have on the system, and built its future wheel model, see Figure 7.

We found that the consequences of the "Successful adoption of Digital Terrestrial TV" would be "More TV channels available", "TV available in mobile devices", "EPG (Electronic Programming Guide) broadcast" and "High Definition". In its turn, "High Definition" has another consequence, "People more likely to watch movies on TV". Then, for each leaf consequence, we identified what would be their consequences to the system, which we call direct consequences.

Another event that we analyzed is "Economic growth", showed in Figure 8. The consequences that we considered relevant to our domain was "Increasing demand for paid TV" and "Access to more Internet bandwidth". "More people watching videos on the Internet" is a secondary consequence of

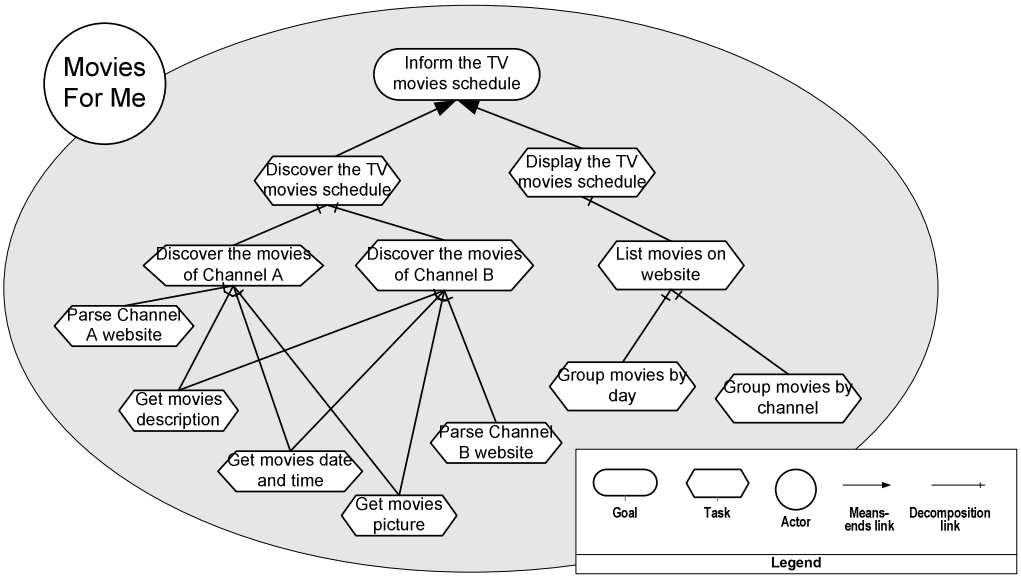


Figure 6 – Initial goal model of the Movies For Me system

this event. Again, for each leaf consequence we identified the direct consequences.

Now, we have to adapt the goal model in such a way that, when those events occur, we do not need to change the system. This adaptation is subjective, and there may be more than a single resulting goal model that deal with those consequences.

Table 2 shows the changes that were made to the goal model, for each direct consequence. To address the consequences A and F, we added the "Add support for a new channel" task. In this way, the system will need to be able to get information from a new channel informed by its user. The "Be search-engine friendly" softgoal was added to address the consequence B, so that when the people search for the TV movies schedule they reach the Movies For Me site. It was also added a softgoal and a task that contributes positively to that softgoal.

With the "Portability" softgoal, we intend to address the consequence C. Moreover, in order to satisfy the "Portability" softgoal, the "List movies on a website" task was further decomposed in three additional tasks, so that each kind of device has a specific, well-suited, website. The consequence E is addressed with the "Get Teaser/Trailer" task, so that the system may provide a video preview of the movies for its users.

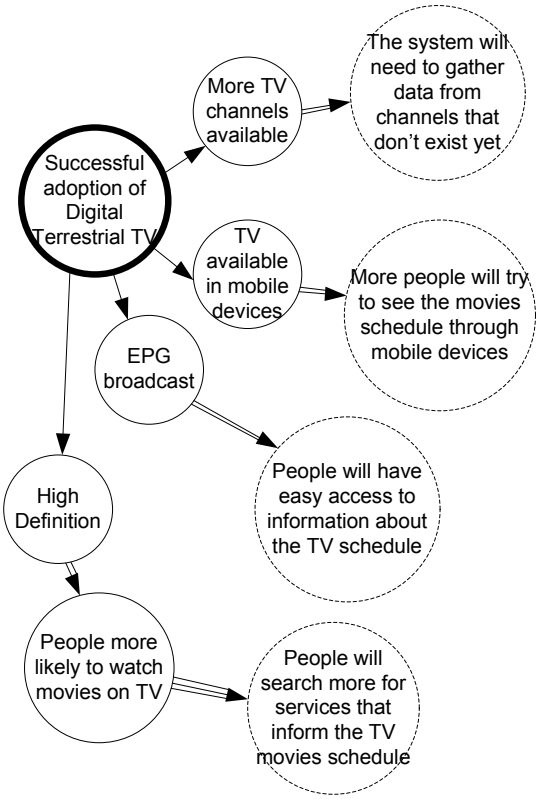


Figure 7 – Futures wheel model for the event "Successful adoption of Digital Terrestrial TV"

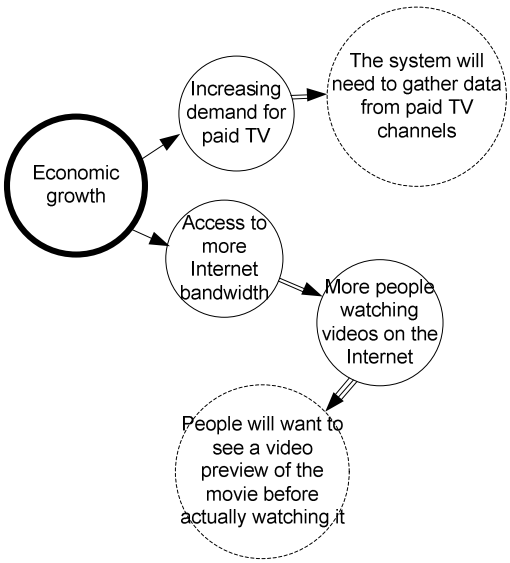


Figure 8 – Futures wheel model for the event "Economic growth"

Despite of consequence D being a direct consequence, we decided that it was not necessary to specifically address it in our goal model, considering that the current model already handles it.

The resulting goal model of the Movies For Me system is presented in Figure 9. Without the analysis on the futures wheel models, these changes would only be made when the system was already developed and implanted, which is more costly than making the changes before the system is developed.

## VII. DISCUSSION

An important tradeoff when anticipating changes is that between the cost of doing it and the cost of not doing it. How costly is it to perform this change now and how costly will it be to perform this change on the future, if it actually becomes required? Moreover, whilst anticipating decisions based on one expected future may be rewarding if this prevision shows to be correct, unnecessary costs may arise if the prevision was not correct. So it is also needed a balance between the costs and the probability of the future change to happen. Regarding this probability, the bigger the time frame used for foresight, the smaller is its accuracy – according to Tonn, Hemrick and Conrad [6] people imagine the future very clearly in a 2 years time frame; somewhat clearly in a 2 to 20 years time frame, and; not very clearly after 20 years.

Kotonya and Sommerville defined six factors that lead to requirements change [16]: (i) requirements errors, conflicts and inconsistencies; (ii) evolving customer/end-user knowledge of the system; (iii) technical, schedule or cost problems; (iv) changing customer priorities; (v) environmental changes and; (vi) organizational changes. The usage of foresight techniques does not reduce requirements changes which occur due to the factors (i), (ii) and (iii). However, it

TABLE II. TRACEABILITY INFORMATION OF THE DIRECT CONSEQUENCES AND THEIR IMPACT ON THE GOAL MODEL

| Direct consequences | Specific Impact |
|---|---|
| (A) The system will need to gather data from channels that don't exist yet | Add "Add support for a new channel" task |
| (B) People will search more for services that inform the TV movies schedule | Add "Be search-engine friendly" softgoal; Add "Use good keywords" softgoal; Add "Use sponsored links" task |
| (C) More people will try to see the movies schedule through mobile devices | Add "Portability" softgoal; Add "Website for desktops and notebooks" task; Add "Specific website for mobile devices" task; Add "Specific website for TV" task |
| (D) People will have easy access to information about the TV schedule | None |
| (E) People will want to see a video preview of the movie before actually watching it | Add "Get Teaser/Trailer" task |
| (F) The system will need to gather data from channels of paid TV | Add "Add support for a new channel" task |

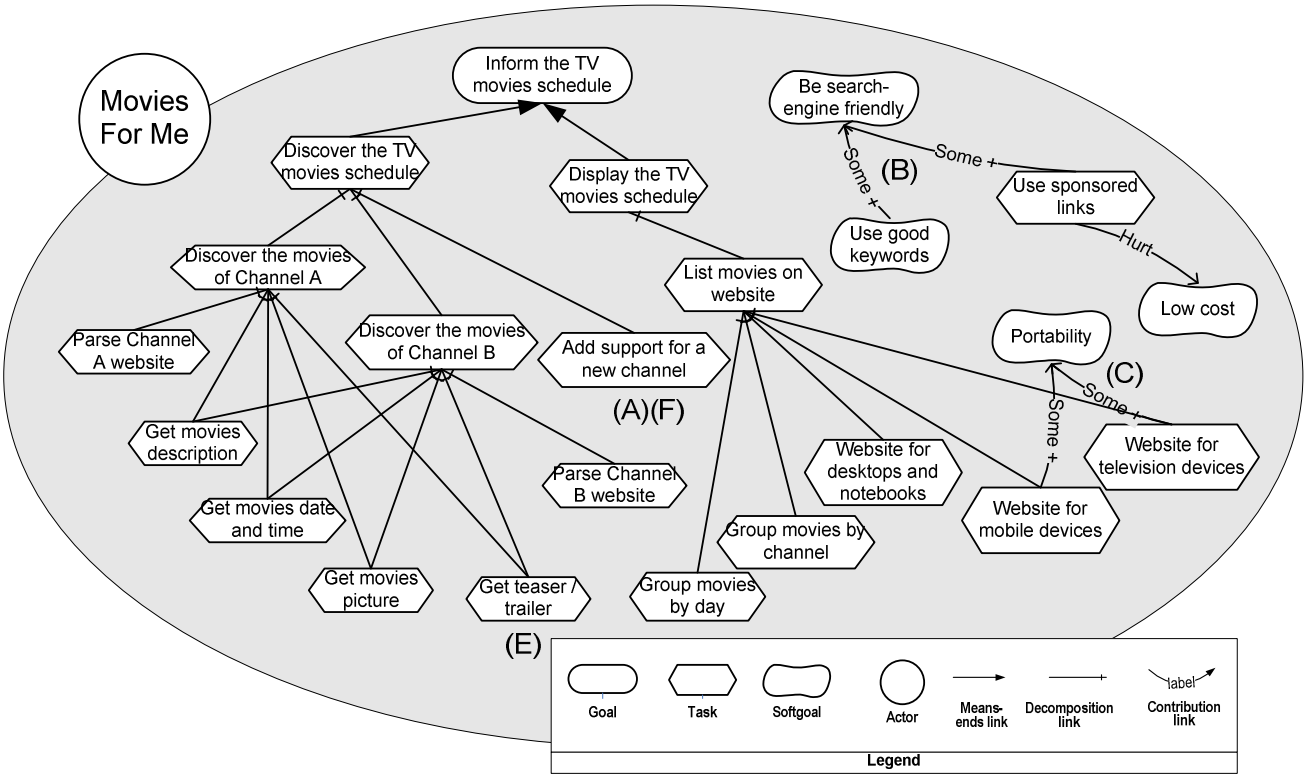


Figure 9 – Final goal model of the Movies For Me system

does have an influence on the last three factors: (iv), (v) and (vi).

There are several works that point out the high cost of changing requirements in later phases of the software development process, such as design or implementation – for instance, [16][28][35]. There are also several works that states that these changes are one of the main causes of software defects or high cost of the software [7][11][18][37][38]. Therefore, by identifying the changes that would be required after the system development, due to a future event, and anticipating these changes, the overall cost of the software development project is likely to be reduced. It is important to note that the constant evolution of Software Engineering techniques and Computer Aided Software Engineering (CASE) tools, the impact of some changes have been significantly reduced. In eXtreme Programming, this ease of modifying software is referred to as an *embrace changes* attitude. However, some kinds of changes still have a large impact on software projects, especially those related to non-functional requirements.

Regarding requirements documentation, there is already an adaptation of use cases for future requirements, called change cases [10]. Further work may need to be performed in order to document future requirements with other requirements description techniques, like goal models or viewpoints. On the approach proposed on this paper, instead of defining future requirements we opted for changing the original requirements model to already incorporate the selected requirements that would arise in the future.

Particularly, studies of the future seem to be very promising on the development of autonomic computing systems and adaptive systems. It may facilitate the implementation of such systems not only during requirements elicitation, but also enabling forecasts performed by the system itself during runtime, based on information from its sensors, as mentioned in [23].

Autonomic computing systems have four main characteristics, which are: self-configuration, self-optimization, self-healing and self-protection [22]. All these four characteristics may be made easier to implement if a representation of the future is used for requirements elicitation. If the system knows how its environment will be in the future, it may be easier for the system to reconfigure to the changed environment (self-configuration). If the system knows how its environment will be in the future, it may be able to make long-term optimizing decisions instead of just short-term decisions (self-optimization). If the system knows some of the problems that it may face in the future, it may be easier for it to take actions to avoid or to correct them (self-healing). Finally, if the system knows that some expected change on its environment may open breach to malicious attacks that it does not suffer yet, it may take actions to protect itself from these attacks (self-protection). Table 3 summarizes these advantages.

If an autonomic system is designed to support a defined space of possible behaviors, like in [2], foresight methods could proof to be a valuable input to their design. A similar situation occurs on (self)-adaptive systems – "Self-adaptive software modifies its own behavior in response to changes in its operating environment" [32]. In most of the approaches, such as [14][29], the changes to which the system may respond to, as well as the responses themselves, need to be defined at design time. To identify these changes, foresight approaches as the one proposed here can be very useful. For instance, to define which components should be adaptable [26] and which failures are more relevant [24][25]. There are also works towards automatically respond to some classes of requirements changes, such as [30][39]. However, these changes are also pre-defined at design time, and could be easier defined using foresight methods.

TABLE III. SUMMARY OF ADVANTAGES OF HAVING A REPRESENTATION OF THE FUTURE, REGARDING AUTONOMIC COMPUTING SYSTEMS MAIN CHARACTERISTICS

| **Characteristic** | **Advantages of having a representation of the future** |
|---|---|
| Self-configuration | Allows early planning of some required adaptations |
| Self-optimization | Allows long-term decisions during runtime |
| Self-healing | Allows early planning on how to deal with some problems |
| Self-protection | Allows early planning on how to deal with some attacks |

## VIII. CONCLUSION AND FUTURE WORK

This paper describes how foresight methods can be included in requirements elicitation to reduce the need of later changes in a software system, whether these changes are manually implemented by human software engineers – on usual systems – or automatically implemented by the system itself – autonomic and adaptive systems. It also describes some foresight methods, selected based on the relevance of their goals to the requirements elicitation process. Finally, it proposes a specific approach to perform changes on requirements expressed through goal models, based on a representation of the future provided by the futures wheel method. This approach is discussed throughout a case study on the context of digital television adoption.

Further research is required to improve the method here presented. For instance, it may be useful to include extra information on the future model, such as associated probabilities of each future event and the estimated cost of preparing the system for them. Moreover, some guidelines to help analysts use the method may be defined – for instance, to decide how to include the new elements in the goal model.

Another future work derived from this paper is the selection of the best-suited foresight methods for using with requirements elicitation, reducing the set of methods presented here. To do so, qualitative researches would be performed to (i) identify how the future environment of a software system is considered on current software development practices, and (ii) evaluate the impact of each foresight method on industrial software projects.

## REFERENCES


[1] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed Requirements Acquisition," Science of Computer Programming, vol. 20, pp. 3-50, 1993.

[2] A. Lapouchnian, Y. Yu, S. Liaskos, and J. Mylopoulos, "Requirements-Driven Design of Autonomic Application Software," Proceedings of the 16th Annual International Conference on Computer Science and Software Engineering CASCON 2006, Toronto, Canada, 2006.

[3] A. Porter et al, "Technology Futures Analysis: Toward Integration of the Field and New Methods," Technological Forecasting & Social Change, vol. 71, pp. 287-303. November 2003.

[4] A. V. Lamsweerde, "Goal-oriented requirements engineering: A guided tour," Proceedings of the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada, pp. 249-262, 2001.

[5] B. Boehm, Software Engineering Economics. Prentice Hall PTR, 1981.

[6] B. Tonn, A. Hemrick, and F. Conrad, "Cognitive representations of the future: Survey results," Futures, vol. 38, issue 7, pp. 810-829, September 2006.

[7] B. W. Boehm and P. N. Papaccio, "Understanding and controlling software costs," IEE Transactions on Software Engineering, vol. 14, pp. 1462-1477, 1988.

[8] D. Loveridge, "Technology Foresight and Models of the Future," Policy Research in Engineering, Science and Technology, September 1996.

[9] E. Baniassad, P. C. Clements, J. Araujo, A. Moreira, A. Rashid, and B. Tekinerdogan, "Discovering early aspects," IEEE Software, vol. 23, issue 1, pp. 61-70, 2006.

[10] E. F. Ecklund, L. M. Delcambre, and M. J. Freiling, "Change cases: use cases that identify future requirements," Proceedings of the 11th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '96), pp. 342-358, October 1996.

[11] E. Oz, "When Professional Standards are Lax, The CONFIRM failure and its Lessons," Communication of the ACM, vol. 37, issue 10, October 1994.

[12] E. Yu, "Modelling Strategic Relationships for Process Reengineering," University of Toronto, 1995.

[13] F. Bachmann and P. Clements, "Variability in Software Product Lines," Technical Report CMU/SEI-2005-TR-012, September 2005.

[14] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "An Architecture for Requirements-Driven Self-reconfiguration," Advanced Information Systems Engineering - LNCS, vol. 5565, pp. 246-260, 2009.

[15] G. J. Tesauro and J. O. Kephart, "Foresight-based pricing algorithms in agent economies," Decision Support Systems, vol. 28, issues 1-2, pp. 49-60, March 2000.

[16] G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques, John Wiley & Sons, 1998.

[17] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne : Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling," Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10), 2010.

[18] I. Navarro, N. Leveson, and K. Lundqvist, "Reducing the Effects of Requirements Changes through System Design,", SERL report, 2000.

[19] J. Glenn, Futures Research Methodology. The United Nations University, USA, 1999.

[20] J. Glenn, "Futurizing Teaching vs Futures Course," Social Science Record, Syracuse University, vol. IX, issue 3, 1972.

[21] J. Heckman and E. Leamer, Handbook of Econometrics, Handbooks in Economics 2, vol. 6A, Elsevier, 2007.

[22] J. Kephart and D. Chess, "The Vision of Autonomic Computing," IEEE Computer, vol. 36, issue 1, pp. 41-50, 2003.

[23] J. Kephart, "Research challenges of autonomic computing," Proceedings of the 27th International Conference on Software Engineering (ICSE'05), pp. 15-22, May 2005

[24] J. Pimentel, E. Santos, J. Castro, "Conditions for ignoring failures based on a requirements model," Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE), San Francisco Bay, USA, pp. 48-53, July 2010.

[25] J. Pimentel, J. Castro, X. Franch, "Specification of Failure-Handling Requirements as Policy Rules on Self-Adaptive Systems," Proceedings of the 14th Workshop on Requirements Engineering (WER 2011), Brazil, 2011, in press.

[26] J. Pimentel, X. Franch, J. Castro, "Measuring Architectural Adaptability in i* Models," Proceedings of the XIV Ibero-American Conference on Software Engineering (CIBSE 2011), Brazil, 2011, in press.

[27] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering, Springer, 2000.

[28] L. Rosenberg and L. Hyatt, "A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality," Product Assurance Symposium and Software Product Assurance Workshop, p. 209, 1996.

[29] M. Morandini, L. Penserini, and A. Perini, "Towards goal-oriented development of self-adaptive systems," Proceedings of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 08), pp. 9–16, USA, 2008.

[30] N. A. Qureshi, A. Perini, N. A. Ernst, and J. Mylopoulos, "Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems," Proceedings of the 1st International Workshop on Requirements at Run-time, Sidney, Australia, 2010.

[31] Object Management Group (OMG), 2009. Unified Modeling Language. (UML) Specification, 2.2. Available at: http://www.omg.org/spec/UML/2.2/.

[32] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An Architecture-Based Approach to Self-Adaptive Software," IEEE Intelligent Systems, vol. 14, pp. 54-62, May 1999.

[33] P. Schwartz, The Art of the Long View, Doubleday, 1991.

[34] S. Boissau and J. C. Castella, "Constructing a common representation of local institutions and land-use systems through simulation gaming and multiagent modeling in rural areas of northern Vietnam: The samba-week methodology," Simulation & Gaming, vol. 34, pp. 342-357, 2003.

[35] S. Ferreira, J. Collofello, D. Shunk, and G. Mackulak, "Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation," Journal of Systems and Software, vol. 82, pp. 1568-1577, 2009.

[36] T. Gordon and J. Glenn, "Integration, Comparisons, and Frontiers of Futures Research Methods," New Technology Foresight, Forecasting and Assessment Methods, Seville, 2004.

[37] T. Javed, M. Maqsood, and Q. S. Durrani, "A study to investigate the impact of requirements instability on software defects," SIGSOFT Software Engineering Notes, vol. 29, pp. 1-7, May 2004.

[38] The Challenges of Complex IT Projects. The report of a working group from The Royal Academy of Engineering and The British Computer Society. Westminster, London, April 2004.

[39] Y. Jian, T. Li, L. Liu, and E. Yu, "Goal-Oriented Requirements Modelling for Running Systems," Proceedings of the 1st International Workshop on Requirements at Run-time, Sidney, Australia, 2010.

[40] Y. Mao, J. Vassileva, and W. Grassmann, "A System Dynamics Approach to Study Virtual Communities," Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS), January 2007.

[41] Y. Yu, J. C. S. D. P. Leite, and J. Mylopoulos, "From goals to aspects: discovering aspects from requirements goal models," Proceedings of the 12th IEEE International Requirements Engineering Conference (RE 2004), pp. 33-42, Kyoto, Japan, 2004.