Anticipating Requirements Changes - Using Futurology in Requirements Elicitation

João Pimentel, Universidade Federal de Pernambuco, Brazil

Emanuel Santos, Universidade Federal de Pernambuco, Brazil

Jaelson Castro, Universidade Federal de Pernambuco, Brazil

Xavier Franch, Universitat Politècnica de Catalunya, Spain

ABSTRACT

It is well known that requirements changes in a later phase of software developments is a major source of software defects and costs. Thus, the need of techniques to control or reduce the amount of changes during software development projects. We advocate the use of foresight methods as a valuable input to requirements elicitation, with the potential to decrease the number of changes that would be required after deployment, by anticipating them. In this paper we define a process for using a foresight method, namely Futures Wheel, for requirements elicitation. To illustrate the use of this approach, we perform a case study using a route planning system.

Keywords: *Requirements elicitation, Requirements changes, Requirements Evolution, Studies of the future, Foresight methods, Autonomic Computing, Self-adaptive systems.*

INTRODUCTION

In the life cycle of a software product, maintenance is considered to be one of the most costly phases (Schach, 2002; Wall & Sinnadurai, 1998). This is largely due to the correction of errors that were introduced in previous phases as well as requirements changes due to the increasingly dynamic context in which the systems run. Moreover, the dynamic business environments and technological improvements lead to the high occurrence of requirements changes. However, requirements evolution may impact other requirements, as well as affect system design, code and test cases. Requirements changes are also one of the main causes of software defects (Javed, Maqsood & Durrani, 2004; Navarro, Leveson & Lundqvist, 2000; Oz, 1994; RAE & BCS, 2004). It has been reported that the sooner a change is detected the better, i.e., the costs for dealing with it are reduced (Rosenberg & Hyatt, 1996). Thus, if we can anticipate these changes during the initial development of the system, we have better chances to minimize their impact on the overall product life cycle.

Nowadays, there is a type of system that is expected to analyze and implement some of these changes at runtime (Lapouchnian, Yu, Liaskos & Mylopoulos, 2006). Indeed, autonomic and self-adaptive systems are able to monitor the environment on which they are running, in order to identify the need for changing their behavior. In order to do so, it is required that these alternative behaviors are previously identified and defined. Therefore, identifying the expected changes in system requirements and defining how to handle these changes is a key research challenge in information systems engineering.

In this paper we claim that the use of foresight methods can provide valuable inputs for requirements elicitation, with the potential of decreasing the number of changes in the software lifecycle. Some works have already shown the benefits of using and adapting well-established methods from social sciences – e.g., ethnography, for requirements elicitation (Neto, Gomes, Castro & Sampaio, 2005). Based on these experiences, we believe that elaborating on the current methods of foresight used by social scientists and futurists is a promising way to predict requirements changes. Thus, in this paper we outline a process based on a specific foresight method – Futures Wheel (Glenn, 1972) – to enrich a requirements model. In order to analyze the suitability of the proposed approach, we performed a case study using a route planning system.

DISCOVERING THE FUTURE

If discovering the current requirements of a system is already a complex task, what to say about the requirements for the future? We can affirm that it is even more challenging, since we may face several cases in which it is impossible to know for sure if an event expected to happen in the future 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 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.

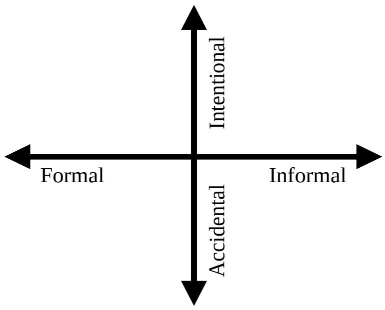
Definition 1 (Future event): *a future event is an event that is expected to take place in the future.*

According to Kotonya and Sommerville (1998), 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 aim at eliciting requirements dealing with future events, we need to consider the projection of these four dimensions in the future. For this purpose, some kind of representation of the future becomes necessary.

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 (Loveridge, 1996). Hence, it may occupy any position on the axes of Figure 1. The best representations of the future would be obtained if it was possible to create a formal and intentional model of the future, but not every project has sufficient resources or knowledge to create such a model. In these cases, the requirements engineer may 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 (Ecklund, Delcambre & Freiling, 1996). This model would be informal, and could be either accidentally or intentionally created.

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



In the literature of future studies, futurology, and foresight there are several techniques and methods that support a rational discovery of possible futures (Glenn, 1999; Porter et al., 2003). These representations of futures may contain just one specific future event, or multiple future events. 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 (Heckman & Leamer, 2007) and Scenarios (Schwartz, 1991), among others.

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

In a previous paper we presented a survey on foresight methods, on which seventeen methods that may be used for requirements elicitation were identified and briefly described: Delphi, Futures Wheel, Participatory methods, Econometrics forecast, Regression Analysis, Trend Impact Analysis, Structural Analysis, System Dynamics, Agent Modeling, Cross Impact Analysis, Relevance Trees, Simulation Modeling, Multiple Perspectives, Causal Layered Analysis, Scenarios, Field Anomaly Relaxation, and Simulation & Gaming (Pimentel, Castro, Perrelli, Santos & Franch, 2011).

There are approaches relating software engineering and some foresight methods, like Delphi (Boehm, 1981), System Dynamics (Mao, Vassileva & Grassmann, 2007), Agent Modeling (Tesauro & Kephart, 2000) and Simulation Gaming (Boissau & Castella, 2003). 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. From the existing foresight methods, we identified Futures Wheel (Glenn, 1972) as a suitable method for requirements elicitation because: (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.

BACKGROUND

In this section we present the Futures Wheel foresight method and its notation for writing representations of the future (Glenn, 1999) which will be used in our requirements elicitation processes. Also, we describe a goal modeling notation that will be used to express system requirements in our case study.

Futures Wheel

In this section we are going to present Futures Wheel according to its standard definition (Glenn, 1999). Futures Wheel is a foresight method that provides a model based on the consequences of a future event or a current trend. This model is a representation of the future. The method is subjective and qualitative, relying on the experience and knowledge of the participants. Its low complexity allows its usage without requiring any specialized training. Nonetheless, it does require a deep understanding of the problem domain being analyzed, so that the generated representation of the future may be as accurate as possible. Therefore, the strong involvement of project stakeholders during the model generation, including client's representatives and domain experts, is a key success factor.

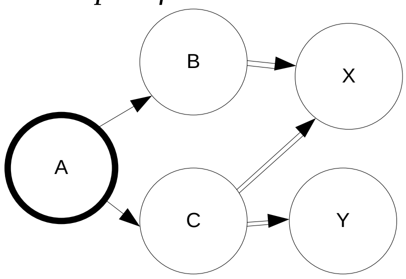
Futures Wheel can be performed either by a single person – e.g., the requirements analyst of a project – or it can be performed collaboratively, usually by means of meetings lead by a mediator. The method itself consists of two steps.

The first step is to identify trends or events that are likely to occur in a near future and that are related to 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*. This naming decision does not imply that a future event has more priority or importance over trends.

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. A leaf consequence is a consequence that has no further consequences.

The application of Futures Wheel creates a representation of the future for each event - a graph in which it is possible to analyze the possible consequences of that event. 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 this notation cannot represent 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



Goal modeling

In goal-oriented approaches (Lamsweerde, 2001), 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 a consequence of the increasing use of goal-orientation in RE, several frameworks, languages and techniques where goals are used as abstraction have emerged, including KAOS (Dardenne, Lamsweerde & Fickas, 1993), the NFR Framework (Chung, Nixon, Yu & Mylopoulos, 2000), *i** (Yu et al., 2010), V-Graph (Yu, Leite & Mylopoulos, 2004) and Techne (Jureta, Borgida, Ernst & Mylopoulos, 2010).

Among these approaches, we chose *i** (Yu et al., 2010), which will be briefly presented in this subsection. Besides being the notation used in the original requirements document of the system considered in our case study, *i** provides a suitable mechanism for representing alternative behaviors of a system, through means-end links. This characteristic makes it more natural to integrate the future-influenced requirements with the current goal model of the system. A future-influenced requirement is a requirement that was created or modified based on a future event.

*i** defines models to describe both the system and its environment in terms of intentional dependencies among strategic actors (Lucena et al., 2008) (*who*). There are two different diagrams, or views, of an *i** model: the Strategic Dependency (SD) diagram presents only the actors and the dependency links amongst them, whilst the Strategic Rationale (SR) diagram 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. 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 are similar to goals, but in this case the 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. Tasks represent a way to perform some activity to obtain satisfaction of a goal or of a softgoal. Resources represent data or information that an actor may provide or receive.

There is one kind of dependency related to each one of the four elements previously defined. A goal dependency states that the depender needs the dependee to satisfy one of its goals. 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). *Means-end* links define 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 *Goal G1*). *Task-decomposition* links describe 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., *Task T2* contributes negatively to *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, in order to perform *Task T3*, *Actor A1* needs *Resource R1* that can be provided by the execution of *Task T4* by *Actor A2*.

FUTURES WHEEL EXTENSION FOR REQUIREMENTS ELICITATION

As noted earlier, a Futures Wheel model describes a future event and its consequences. Naturally, there is still a large gap between the consequences and the system requirements. In order to diminish this gap, we have enlarged the notation with a new type of consequence, that we name *direct consequence*. Direct consequences act as a layer between regular consequences and system requirements.

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

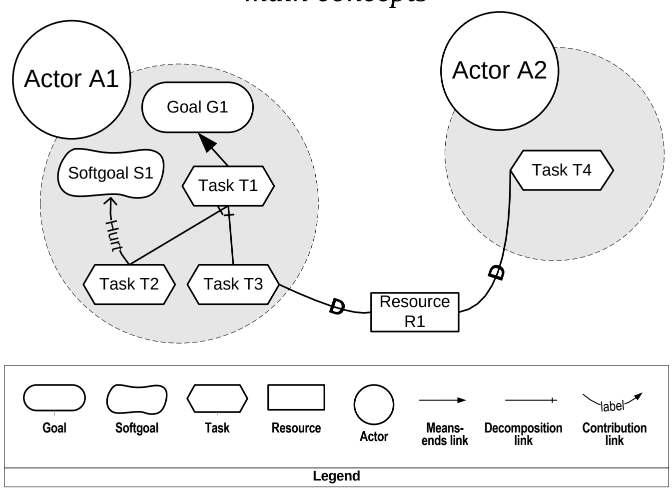
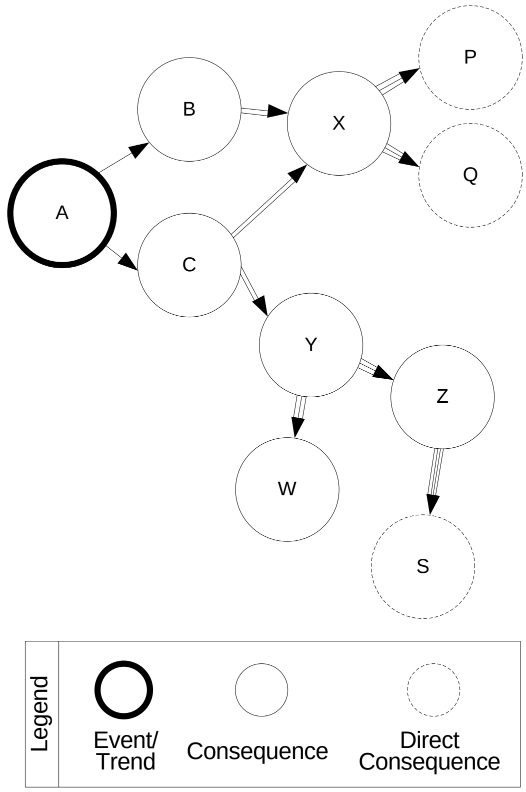


Figure 4 – Example of the extended Futures Wheel model notation



Hence, for each regular consequence, we may ask "how does this consequence affect the system"? I.e., what kind of direct system support (service, operation, function) is required? The answer will be one or more direct consequences, since they are directly related to the system. To make it explicit which are the direct consequences, we represent them as circles with a dashed border.

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.

The metamodel of this extended Futures Wheel notation is presented in Figure 5, using the Unified Modeling Language - UML (OMG, 2009b). 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 at each model. An event may have an indefinite number of consequences. There are two types of consequence: regular consequence and direct consequence. Each consequence must have at least one source, either an event or a regular consequence. A regular consequence may have any number of (sub-)consequences.

Both the Event class and the Consequence class have an attribute to represent the description of their instances – for instance, to describe what is the future event being modeled. This attribute is inherited by the Regular Consequence and Direct Consequence classes. The Event class has two additional attributes: Timeframe, that states when that event is expected to take place; and Probability, that indicates the likelihood of that event to happen.

We decided not to include attributes such as priority, source, impact, and others, since this would depend on the requirements process and templates being used.

FUTURES WHEEL FOR REQUIREMENTS ELICITATION – THE PROCESS

We now present our process to guide the use of the Futures Wheel method for requirements elicitation. It was designed to be deployed in concert with some other current requirements engineering process. Hence, the process does not restrict the elicitation techniques to be used, neither the requirements models to be created, and so on.

Figure 6 outlines the Futures Wheel for Requirements Elicitation Process. The inputs are the Requirements Document and the possible templates that the organization may already have for using Futures Wheel (Futures Wheel Plan Template and Futures Wheel Document Template). The output of the process is the Futures Wheel Document, which may contain the Futures Wheel models and additional descriptions of the models.

Figure 5 – Metamodel of the extended Futures Wheel notation

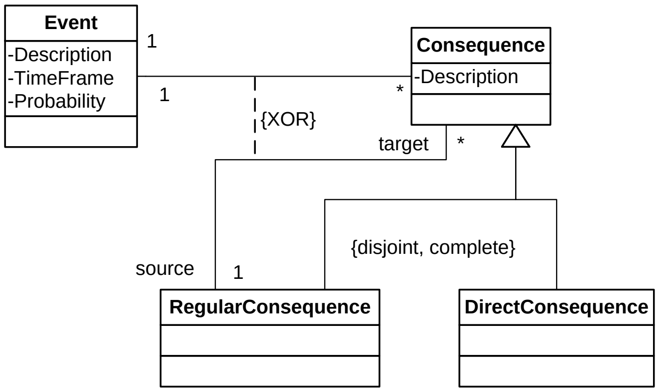
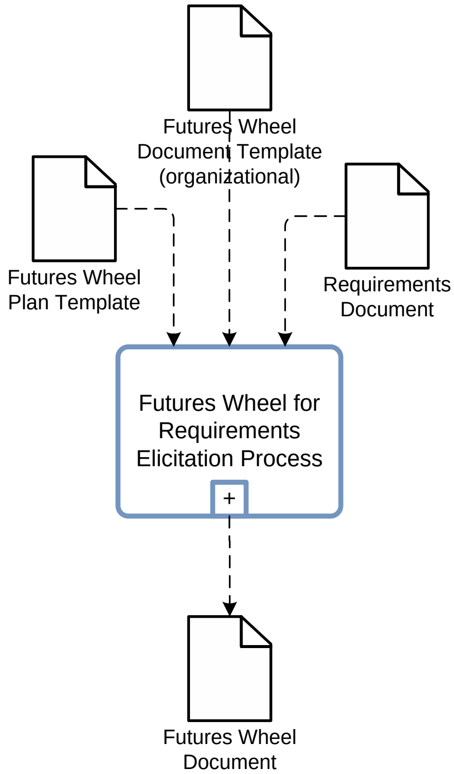


Figure 6 – Inputs and Outputs of the Futures Wheel for Requirements Elicitation Process



The process comprises four activities (Figure 7): Plan Futures Wheel, Perform Futures Wheel, Define Direct Consequences, and Analyze Direct Consequences. These activities will be described in the following sub-sections.

Both figures 6 and 7 present the process using the Business Process Model and Notation – BPMN (OMG, 2009)

Plan Futures Wheel

This activity consists of defining how the Futures Wheel method is going to be deployed in the specific project under consideration. If the organization has already adopted a Futures Wheel Plan Template, it can be used to guide this planning, for management purpose. Similarly, if it also has a Futures Wheel Document template, in this activity it will be instantiated to the specific project being carried on. This instantiated document will contain the Futures Wheel models generated throughout the process.

The Futures Wheel Document Template may include, among others, the following sections: History Control, Index, Scope, Timeframe, Futures Wheels Models, Futures Wheels Descriptions, Assumptions, and Glossary.

The template for Futures Wheel Document may contain usual project plan sections, such as History Control, Index, Scope, Stakeholders, Resources, Schedule, Budget, Risks, Change Control, Work Breakdown Structure, Assumptions, and Glossary. The 5W2H (What, Why, Where, When, Who, How and How much) dimensions can be used to guide the planning for carrying the Futures Wheel method:

What – What tasks are going to be realized? For instance: Interviews, Questionnaires, Focus groups, Reviews.

Why – What is the rationale for each task to be realized?

How – How each task is going to be performed? For instance, how are conflicts going to be solved during the focus group? Is there going to be a facilitator? Will the meeting be recorded? Will the meeting be held in-person or through the Internet? And so on.

Who – Who is going to be involved in each task? What are their roles? For instance, in a focus group, who are the participants and who is going to be a facilitator?

Where – Where are the tasks going to take place?

When – What is the schedule for performing the process?

How much – How much will it cost to perform these tasks?

The planning may have different degrees of details, according to the size of the developing organization and to the complexity of the particular project being developed. It is important to note that additional information about the organization and about the project being carried can be useful to this planning activity. Such information includes the organization size, structure, resources, the project duration, the requirements techniques being used, and so on.

The outputs of this activity are a Futures Wheel Document Template (instantiated for the project) and a Futures Wheel Plan.

Perform Futures Wheel

After the planning, the Futures Wheel itself can be performed. This activity consists of creating the Futures Wheel models, which results in a representation of the future with events and consequences that may have some impact in the system to be developed. These models are documented in the Futures Wheel Document.

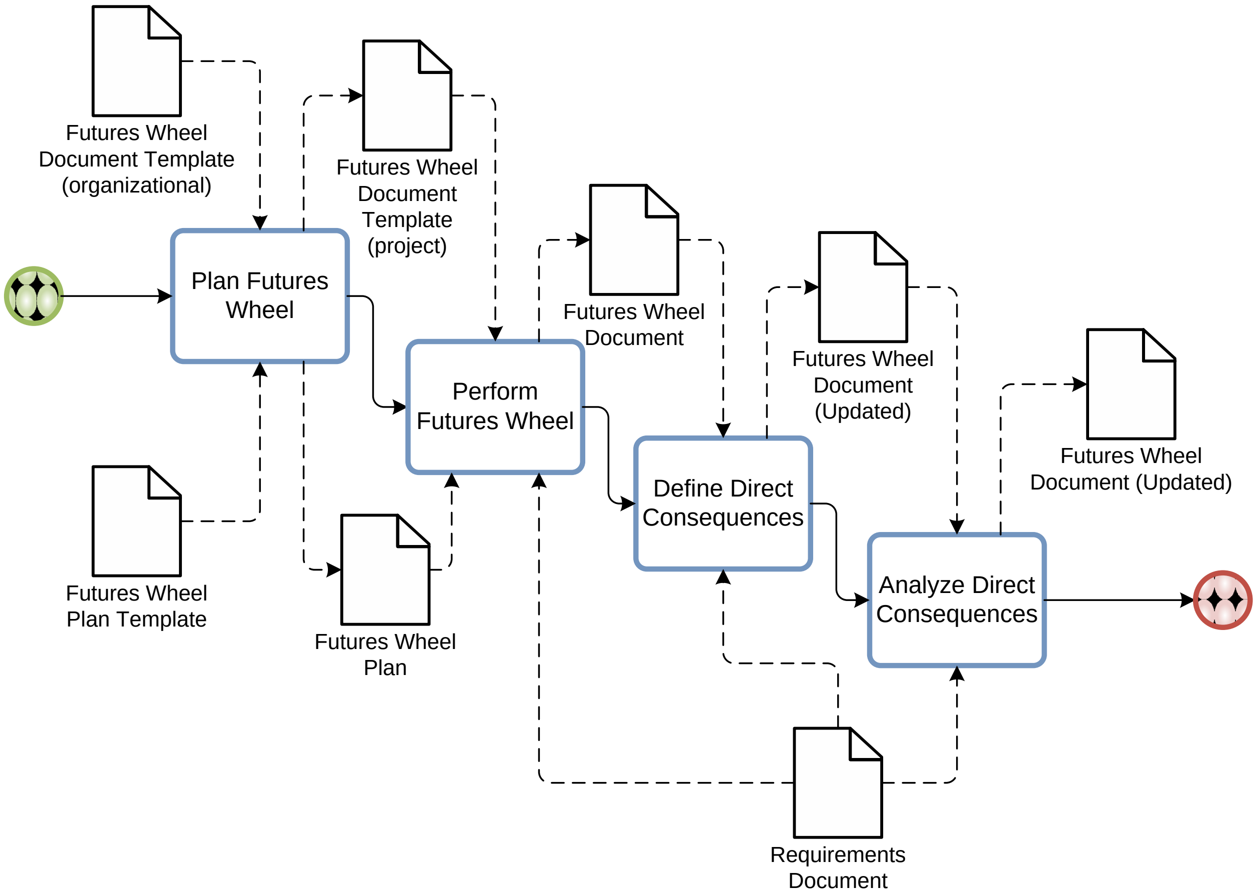
Each event will be identified, as well as the consequences for each event, the consequences of each consequence, and so on, as described in previous sections. When doing so, it is important to consider the system in focus, whether by an informal description or by a brief analysis of its already elicited requirements – which justifies considering the Requirements Document as input for this activity. Otherwise, there would be the risk of identifying too many future events and consequences that are not related at all with the system. Nonetheless, it is important to not restrain too much the modeling to the system requirements, since this could prevent the creation of richer and more useful models.

Creating the Futures Wheel models is a matter of information elicitation. Thus, usual techniques, such as interviews, questionnaires, and focus group, can be used. Moreover, the same good practices and guidelines for requirements elicitation in general can be considered, such as the ones proposed by Sommerville & Sawyer (1997). The decision on how to create these models is taken in the previous activity (Plan Futures Wheel), being described in the Futures Wheel Plan. For specific guidance on creating Futures Wheel models, please refer to the *Background - Futures Wheel* section in this paper. Additional information is also available in Glenn (1999).

When identifying the consequences, we should consider the four requirements elicitation dimensions presented in (Kotonya & Sommerville, 1998): Application domain, Problem to be solved, Business context and Stakeholder needs and constraints.

The output of this activity is a Futures Wheel Document.

Figure 7 – Process for using Futures Wheel for Requirements Elicitation



Define Direct Consequences

This activity has a stronger focus on the desired system, rather than the future scenario. Besides the Futures Wheel Document, the Requirements Document of the system under development is also an input to this activity. Based on these two input documents, the direct consequences will be defined and included in the Futures Wheel Document. A direct consequence describes how a consequence in the Futures Wheel model affects the system being developed. During this activity, the participants may also identify other consequences that will help them to more clearly define the direct consequences. Additionally, if a regular consequence in the model is identified as being a direct consequence, it just has to be stated as so (by changing its border to a dashed one).

Usually direct consequences will be identified from leaf consequences, but this is not mandatory. Even so, it is not expected that every leaf consequence will have a direct consequence. Anyhow, the participants of this activity should keep in mind that defining a direct consequence by no means declares a commitment into actually incorporating that consequence in the system. Further analysis may be needed.

The output of this activity is an updated version of the Futures Wheel Document, including the new direct consequences and other document changes.

Analyze Direct Consequences

The inputs of this activity are the Futures Wheel Document – with the direct consequences – and the Requirements Document of the information system being developed.

In this activity, the direct consequences will be analyzed, in order to determine whether they should be actually considered in the requirements process. This analysis is performed considering the probability of those consequences, their impact, if they are within the project scope, among other factors. 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 to support that consequence. For example, 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.

Additionally, it is important to detect and handle contradictory or conflicting consequences. However, it is important to note that we are not dealing with certainties, but rather with probabilities. Thus, in some cases it may be useful to maintain both conflicting consequences, as long as the conflict is described in the Futures Wheel Document.

After the analysis, the direct consequences are either confirmed or dropped. The revised document will be considered an additional input for the requirements engineering process used by the developing organization. This may result in creating new requirements or in changing already existing requirements. Note that the requirements refinement, prioritization, analysis, and so forth, can be performed as usual in that organization.

In the next section we present a case study that uses the Futures Wheel for Requirements Elicitation process to refine a requirements model.

CASE STUDY

In order to analyze the suitability and exemplify the usage of our approach, we developed a case study based on the By The Way – UFPE (BTW-UFPE) system. This system was developed for the SCORE contest, a software engineering competition held at the 31st International Conference on Software Engineering (ICSE) in 2009. We chose this project because it is a real case study that resulted in an awarded software system. Moreover the produced *i** models are of moderate complexity.

The system itself consisted on a route-planning system that helps users through advices about a specific route searched by the user. This information is posted by other users and might be filtered to provide for the user only relevant information about the place that he/she intends to visit. It was targeted for people that are going to travel to a city and need not only to find out routes to move throughout the city, but also to know additional info based on that route – for instance, entertainment places near the route, or accessibility info on the streets of the route.

The BTW-UFPE project used a process based on the Tropos method (Mylopoulos, Castro & Kolp, 2000) process, consisting of the following disciplines: early requirements, late requirements, architectural design, detailed design, implementation, verification and project management. For requirements elicitation, the following techniques were used: literature analysis, interviews, competitor analysis and prototyping. The *i** models were created using the Process Reengineering *i** Methodology (PR*i*M) (Grau, Franch & Maiden, 2005). Originally, no foresight methods were used.

We describe next the original requirements model of the BTW-UFPE system (*Figure 8*), which will be used throughout this case study. Then, we present the step-by-step application of Futures Wheel for requirement elicitation – i.e., the case study itself. In the sequel, we present the results of the case study. Later on, in the Discussion section, we present further considerations on this case study.

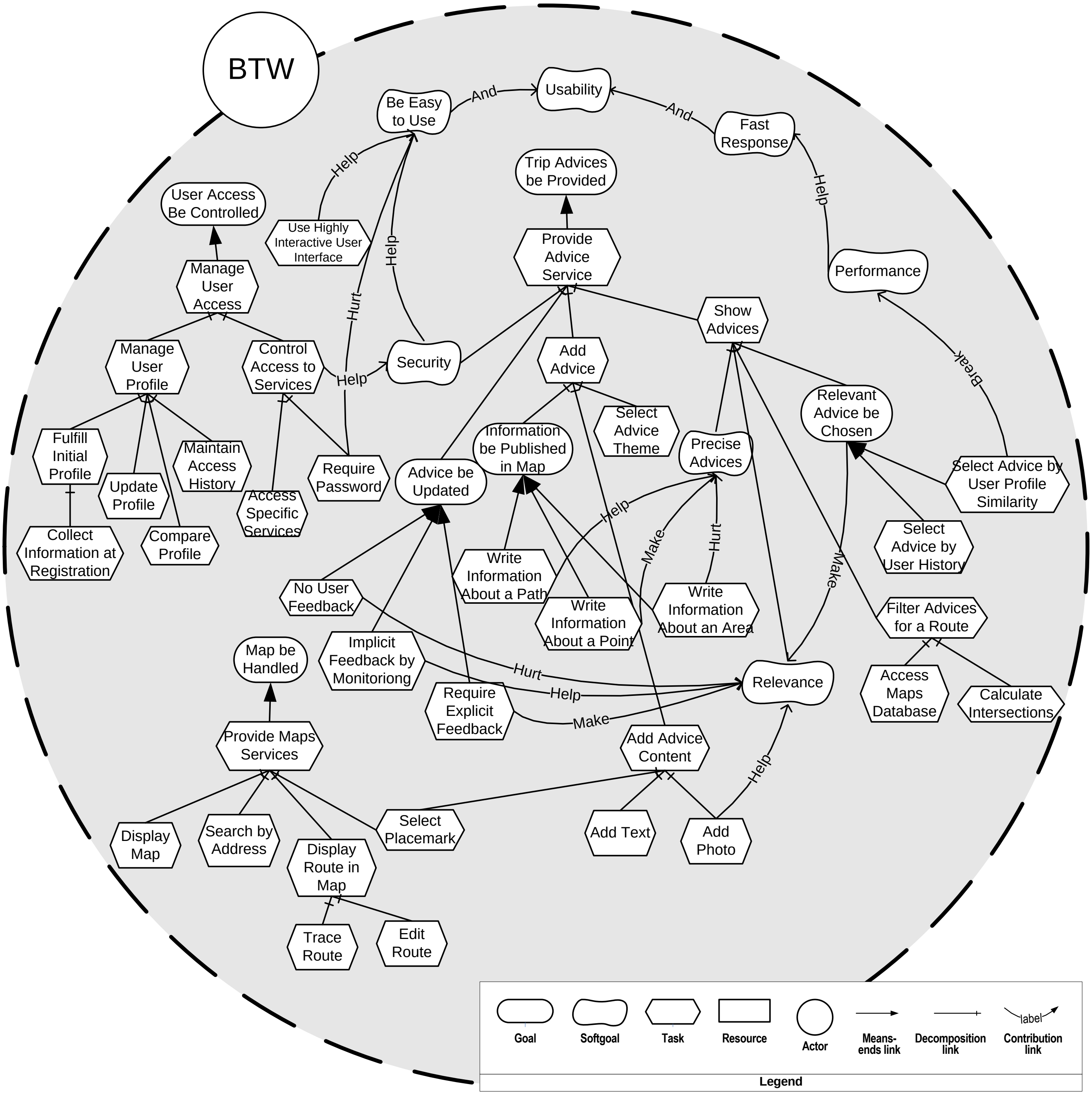
Original Requirements

The requirements document is an input of the Futures Wheel for Requirements Elicitation process, used in its two last activities. For the sake of this case study, we are going to consider the BTW-UFPE system *i** model as being its requirements document. The actual document contains extra information, such as Detailed Interaction Scenarios and Assumptions. As a result of the Futures Wheel for Requirements Elicitation process, this model will be modified. Thus, in the remainder of this subsection we describe *i** model of the BTW-UFPE system.

Figure 8 is an *i** model showing the BTW actor and its internal details, as presented at (Castro et al., in press). For the sake of simplicity, we are omitting the dependencies related to this actor.

The BTW actor represents the software system to be developed. Its main functional goal is *Trip Advices be Provided*. Throughout the refinement of this main goal, other two major goals were identified: *User Access Be Controlled*, and *Map be Handled*. Each one of these goals can be achieved through these respective tasks: *Provide Advice Service*, *Manage User Access* and *Provide Maps Services*. These tasks represent the main functionalities of the system. The decomposition of the *Provide Advice Service* task onto the *Security* softgoal means that this task shall be performed securely. It is further decomposed onto the *Advice be Updated* goal, the *Add Advice* task, and the *Show*

Figure 8 – Initial goal model of the BTW-UFPE system.



Advices task. There are three means to fulfill the *Advice be Updated* goal: without user feedback, with implicit feedback (by monitoring) or requiring explicit feedback (from the users).

The *Add Advice* task consists of publishing the information in a map, as well as in adding advice content, and in selecting the advice theme (i.e., its category). The information can be related to a path in a map (for instance, information about a street), to a point in a map (for instance, a specific restaurant) or to an area in a map (for instance, a car parking area). These alternatives may have different impacts on how precise the advices are, which is a constraint of the *Show Advices* task. *Relevance* is another softgoal of interest. The *Relevance* of the advices is affected by the way advices are updated, as well as by the content that is added: Text and Photo. Lastly, it is also influenced by the *Relevant Advice be Chosen* goal. This goal might be achieved by selecting advices to be shown either by the user history or by users profile similarity. This last option has an impact on *Performance*, which contributes to *Fast Response*. In its turn, the *Fast Response* softgoal impacts *Usability*. In order to perform the *Show Advices* task, it is also required to *Filter Advices for a Route*, i.e., to select only the advices related to a route being searched. In order to do so, it is necessary to perform the *Access Maps Database* and *Calculate Intersections* tasks.

The *Provide Maps Services* task is decomposed onto four tasks: *Display Map*, *Search by Address*, *Display Route in Map* and *Select Placemark*. The *Trace Route* and *Edit Route* tasks are the decomposition of the *Display Route in Map* task.

Related to *Security*, the *Control Access to Services* task is a decomposition of the *Manage User Access* task. *Control Access to Services* is decomposed onto *Access Specific Services* and *Require Password*. The later has an impact on the *Be Easy to Use* softgoal, which impacts *Usability*. The *Use Highly Interactive User Interface* task also impacts the *Be Easy to Use* softgoal. Lastly, the *Manage User Profile* task is decomposed onto *Maintain Access History*, *Compare Profile*, *Update Profile*, and *Fulfill Initial Profile*. The later consists of collecting information at registration.

Using the Futures Wheel Process for Requirements Elicitation

The first activity of the Futures Wheel for Requirements Elicitation process (depicted in Figure 7) is the planning. We used the 5W2H technique to guide the planning, which was documented in a simple Futures Wheel Plan. Then, we designed the Futures Wheel Document Template that we would use in this case study.

In order to perform the other activities of the process, we invited three researchers not related to this paper. For each one of these volunteers, a work meeting was held. At these meetings, the BTW-UFPE system and the Futures Wheel method were briefly described by a facilitator, in 15 to 20 minutes presentations. The facilitator was an author of this paper. The participants were already familiar with *i**, but not with Futures Wheel. Afterwards, in the same meeting, the volunteers were asked to create Futures Wheel models with the help of the facilitator.

When the Futures Wheel models were created (Activity 2), we asked the invited researchers to identify the direct consequences (Activity 3). Note that the *i** model depicted in Figure 8 was an input for this activity. Table 1 presents some measurements taken on the resulting models – the number of events, consequences, leaf consequences and direct consequences defined by each invited researcher. This data showed some correlations that will be further explored in the Discussion section.

Table 1 – Measurements taken on the Futures Wheel models created during the case study

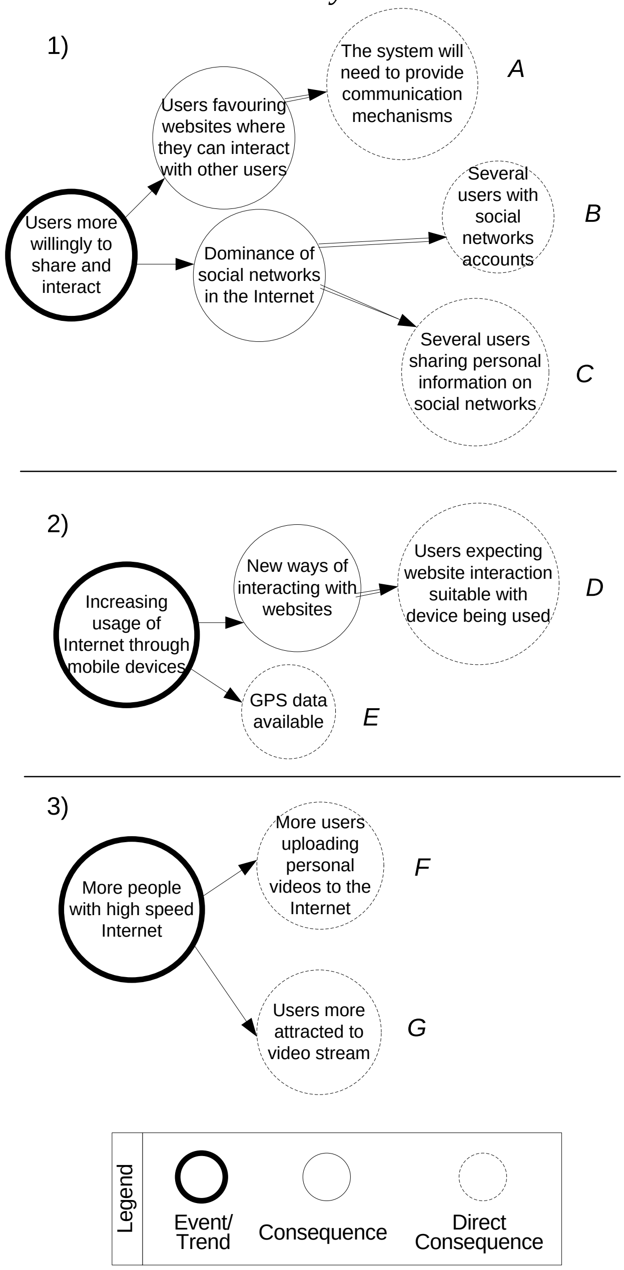
Author	Events	Consequences	Leaf consequences	Direct consequences
XX	5	22	18	9
XY	6	34	17	14
XZ	3	51	25	13

After the creation of three different sets of Futures Wheel models – one by each volunteer – the facilitator analyzed these models. Based on this analysis, the facilitator generated a consolidated model, merging the different models when there were similar events. Afterwards, the Futures Wheel document that contains these models was validated by the same group of volunteers, individually.

The last activity consists in analyzing the direct consequences described in the Futures Wheel Document. This was performed by the facilitator, along with other authors of this paper. In Figure 9 we present an excerpt of the resulting Futures Wheel models, including three events. The first one is the increasing willingness of users to share information and to interact with other users. The second event is about the widespread use of mobile devices, such as smartphones and tablets, to access Internet websites. The last event is related to the availability of better network infrastructure (i.e., faster Internet connections). All these events were defined considering a timeframe of 5 years, which was the timeframe decided during the first activity.

From these three events, seven direct consequences were defined (Figure 9). The first one is related to communication mechanisms (A); the second one is related to social networks accounts (B); the third one is about sharing personal information on social networks (C); the fourth consequence is about user interaction with specific devices (D); the fifth one is about the use of GPS (E); the sixth and the seventh one are related to videos on the Internet (F and G).

Figure 9 – Futures wheel models for the BTW-UFPE system



Results

Considering the direct consequences presented in Figure 9, we analyzed the goal model of Figure 8 in order to identify how it could be modified in order to properly address the consequences.

Table 2 shows the changes that were made to the goal model, for each direct consequence. Addressing consequence A, we included the functionalities of chat and comments. To support consequence B, we added the option to login using a social network account. Consequence C was addressed with new options to publish advices and to share advices in social networks. Consequence D resulted in a new requirement of porting the system to specific devices. To address consequence E we included the option to search address by user position provided by a GPS. Lastly, consequences F and G were taken care of with the option of adding video when adding advices.

The resulting goal model of the BTW-UFPE system is presented in Figure 10. Without the analysis on the Futures Wheel models, these changes would only be made when the system was already developed and released in production, which more costly than making the changes before the system is actually developed.

DISCUSSION

An important tradeoff when anticipating changes is that between the cost of doing it and the cost of not doing it. Hence, the issue to be discussed is the cost to perform these changes now versus delaying them to the appropriate moment. We already know that some technological changes may have dire consequence, for example, in a system's architecture, causing even the system to be totally redeveloped. 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 (2006) people imagine the future very clearly in a 2 years' timeframe; somewhat clearly in a 2 to 20 years' timeframe, and; not very clearly after 20 years.

Note that Kotonya and Sommerville (1998) defined six factors that lead to requirements change: (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 related to factors (i), (ii) and (iii). However, it 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, (Ferreira, Collofello, Shunk & Mackulak, 2009; Rosenberg & Hyatt, 1996). Moreover, changes are one of the main causes of software defects or high cost of the software (Boehm & Papaccio, 1988; Javed, Maqsood & Durrani, 2004; Navarro, Leveson & Lundqvist, 2000; Oz, 1994; RAE & BCS, 2004). Therefore, we hope that foresight methods can help to identify changes that would be required after the

Table 2 - Traceability information of the direct consequences and their impact on the goal model

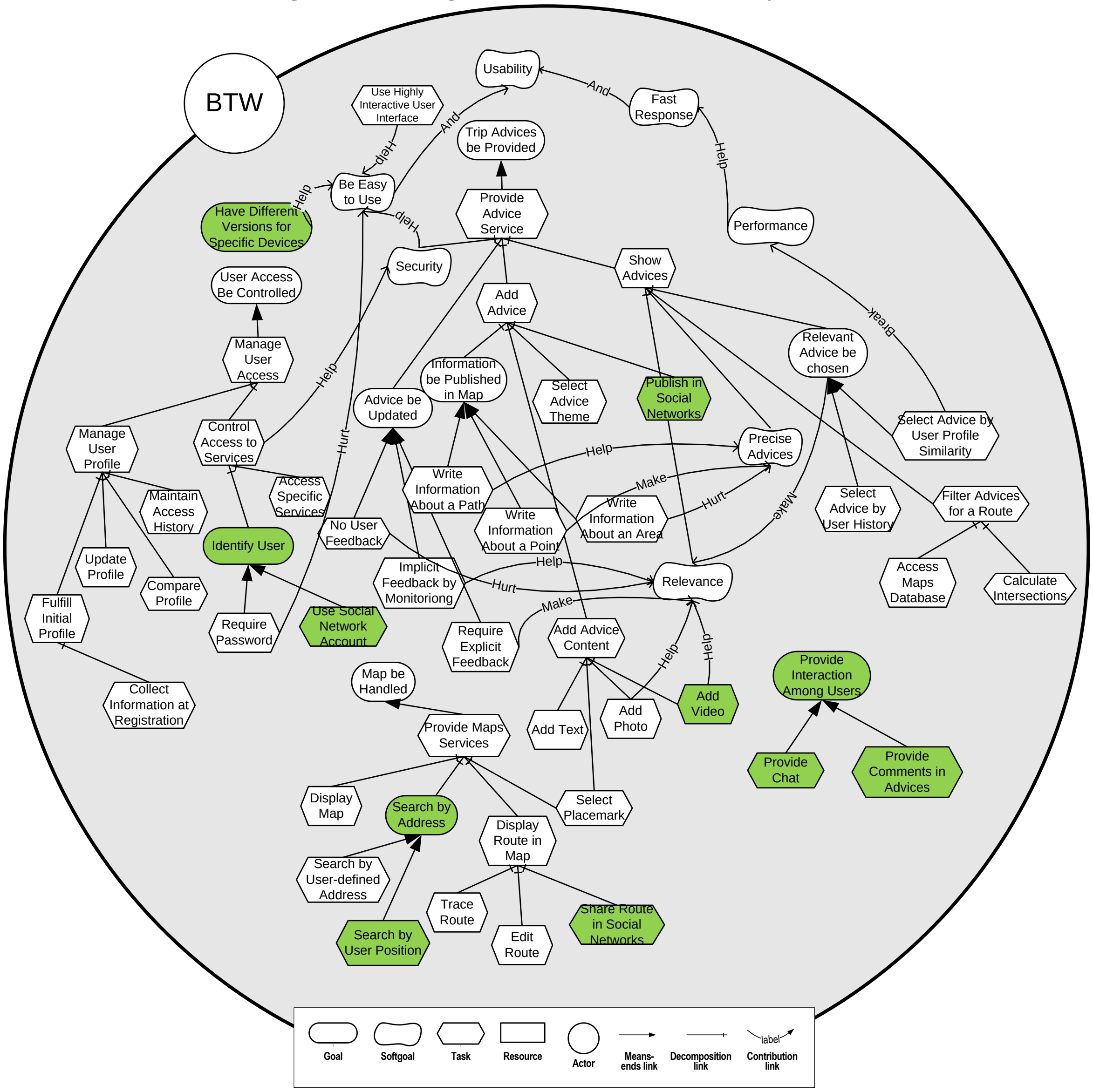
Direct consequences	Specific Impact
(A) The system will need to provide communication mechanisms	Add "Provide Interaction Among Users" goal, with the following means: "Provide Chat" and "Provide Comments in Advices"
(B) Several users with social network accounts	Add "Identify User" goal; Remove task decomposition from "Control Access to Services" to "Require Password"; Add means-end link from "Require Password" to "Identify User"; Add "Use Social Network Account" as a means to "Identify User"
(C) Several users sharing personal information on social networks	Add "Share Route in Social Networks" task, with decomposition link from "Display Route in Map"
(D) Users expecting website interaction suitable with device being used	Add "Have Different Versions for Specific Devices" goal, with Help contribution to "Be easy to use"
(E) GPS data available	Remove "Search by Address" task; Add "Search by Address" goal, with the following means: "Search by User-defined Address" and "Search by User Position"
(F) More users uploading personal videos to the Internet	Add "Add Video" task, with a decomposition link from "Add Advice Content" and a Help contribution link to "Relevance"
(G) Users more attracted to video stream	Add "Add Video" task, with a decomposition link from "Add Advice Content" and a Help contribution link to "Relevance"

system development (due to a future event). This may help to reduce the overall maintenance cost. The representations of the future may also help release planning and affect other project decisions, such as whether to develop families of system or just a single system.

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 (Ecklund, Delcambre & Freiling, 1996). Further work may need to be performed in order to document future requirements with other requirements description techniques, such as goal models or viewpoints. In this paper, instead of defining a new notation include the future requirements, we opted for changing the original requirements model to incorporate the selected requirements that would arise in the future. This changing may be performed by provoking (i) the creation of new requirements; (ii) the exclusion of requirements that already exist;

Figure 10 – Final goal model of the BTW-UFPE system



(iii) changes on requirements that already exist; or (iv) by combining any of these three types of change.

Case Study

An analysis on the measurements on the Futures Wheel models created during the case study (Table 1) suggests that metrics could be created to evaluate some characteristics of these models. The ratio of leaf consequences per regular consequences may express the degree of details of the model. For example, in the given case study we could observe the following correlation: the lowest this ratio, the higher the details.

We may express this metric in OCL (OMG, 2010), considering the metamodel depicted in Figure 5. In order to do so, we will need to calculate the number of regular consequences in this model. This can be achieved with the following expression:

```
AmountOfRegularConsequences ::=
RegularConsequence.allInstances()->size()
```

. The number of leaf consequences, ignoring direct consequences, is the number of regular consequences that do not have any other regular consequence as target. Thus,

```
AmountOfLeafConsequences ::=
RegularConsequence.allInstances()->select(e|e.target-
>select(d|d.oclIsTypeOf(RegularConsequence))->size()
= 0)->size()
```

. Hence, to calculate the given ratio, we just have to divide the amount of leaf consequences (not considering direct consequences) by the amount of regular consequences:

```
Metric1 ::= RegularConsequence.allInstances()-
>select(e|e.target-
>select(d|d.oclIsTypeOf(RegularConsequence))->size()
= 0)->size()  /  RegularConsequence.allInstances()-
>size()
```

Another possible metric is the ratio between direct consequences and leaf consequences. The empirical evaluation of the models created during the BTW-UFPE case study showed that the models with a lower value in this ratio are less focused and somewhat less useful. To express this metric using OCL, we need to calculate the amount of direct consequences in the model:

```
AmountOfDirectConsequences ::=
DirectConsequence.allInstances()->size()
```

Dividing the number of direct consequences by the number of leaf consequences, we have a second metric:

```
Metric2 ::= DirectConsequence.allInstances()->size()  /
RegularConsequence.allInstances()->select(e|e.target-
>select(d|d.oclIsTypeOf(RegularConsequence))->size()
= 0)->size()
```

A third metric that emerged from the analysis of the case study is counting how many incoming arrows a direct consequence has. I.e., a direct consequence is a sub-consequence (target) of how many consequences. We expect that higher the value of this metric, the more important the given direct consequence may be, assuming equal priorities. However, we were not able to establish such a correlation based on the data of our case study.

Expressing this metric in OCL, for a given direct consequence, we have the following expression:

```
context DirectConsequence
Metric3 ::= self.source->size()
```

Of course, the above metrics are only a first attempt to develop metrics for futures wheel models. Further experiments need to be conducted in order to better understand and validate these metrics.

Autonomic Computing

Particularly, studies of the future seem to be very promising for 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 (Kephart, 2005).

Recall that Autonomic computing systems have four main characteristics, which are: self-configuration, self-optimization, self-healing and self-protection (Kephart & Chess, 2003). 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, the system reconfiguration to adapt to the environment may be facilitated, enabling 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 can predict 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 did 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 (Lapouchnian, Yu, Liaskos & Mylopoulos, 2006; Santos et al., 2011) foresight methods could proof to be an invaluable input to their design. A similar situation occurs on (self)-adaptive systems, which modify their own behavior in response to changes in its operating environment (Oreizy et al., 1999). In most of the approaches, such as (Franch et al., 2011; Morandini, Penserini & Perini, 2008; Pimentel et al., in press), the changes to which the system may respond to, as well as the responses themselves, need to be defined at design time. Foresight approaches as the one proposed here can be very useful to identify these changes. For instance, they may be deployed to define which components should be adaptable (Pimentel, Franch & Castro, 2011) as well as to identify the most relevant failures (Pimentel, Santos & Castro, 2010). There are also works towards automatically responding to some classes of requirements changes, such as (Jian, Li, Liu & Yu, 2010; Qureshi, Perini, Ernst & Mylopoulos, 2010). However, these changes are also pre-defined at design time, and could benefit of foresight methods.

Table 3 - 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

CONCLUSIONS AND FUTURE WORK

This paper presented a process for using a foresight method for requirements elicitation – in particular, the Futures Wheel method. The proposed process comprises four steps: Plan Futures Wheel, Perform Futures Wheel, Define Direct Consequences and Analyze Direct Consequences. Moreover, an extension of the Futures Wheel modeling notation and of the method itself was presented, aiming to make them more suitable for requirements elicitation. The process was design allowing for its use in conjunction with other requirements techniques, models and processes.

In order to analyze the suitability of the proposed approach, a case study was performed in the domain of route planning. This study proved the concept and showed that, for this particular case, the approach provided more inputs for requirements elicitation, which in its turn provided a richer requirements model. Further research is required to evaluate the usefulness of the proposed approach, as well as of the metrics that were identified during the case study. Additionally, it would be interesting to provide more formalized guidance rules for creating and analyzing Futures Wheel models

Other than that, we intend to perform a thorough analysis on how foresight methods can be used in the development of autonomic systems. This includes analyzing other foresight methods, as those presented in (Pimentel, Castro, Perrelli, Santos & Franch, 2011). Lastly, we intend to investigate the possibility of using foresight methods in other software engineering disciplines, such as architectural design and system testing.

ACKNOWLEDGMENTS

We are thankful to the volunteers who participated in the case study. This work was partially supported by CAPES, CNPq, and the Spanish research project TIN2010-19130-c02-01

REFERENCES

Boehm, B., & Papaccio, P. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, 14(10), 1462-1477.

Boehm, B. (1981). *Software Engineering Economics*. Prentice Hall PTR.

Boissau, S., & Castella, J. (2003). 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*, 34(3), 342-357.

Castro, J., Lucena, M., Silva, C., Alencar, F., Santos, E., & Pimentel, J. (in press). Changing attitudes towards the generation of architectural models. *Journal of Systems and Software*.

Chung, L., Nixon, B., Yu, E., & Mylopoulos, J. (1999). *Non-Functional Requirements in Software Engineering*. Springer.

Dardenne, A., Lamsweerde, A., & Fickas, S. (1993). Goal-directed Requirements Acquisition. *Science of Computer Programming*, 20(1-2), 3-50.

Ecklund, E., Delcambre, L., & Freiling, M. (1996). Change cases: use cases that identify future requirements. In *11th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications – OOPSLA* (pp. 342-358).

Ferreira, S., Collofello, J., Shunk, D., & Mackulak, G. (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. *Journal of Systems and Software*, 82(10), 1568-1577.

Franch, X., Grünbacher, P., Oriol, M., Burgstaller, B., Dhungana, D., López, L., Marco, J., & Pimentel, J. (2011). Goal-driven Adaptation of Service-Based Systems from Runtime Monitoring Data. In *5th IEEE International Workshop on Requirements Engineering For Services - REFS*.

Glenn, J. (1972). Futurizing Teaching vs Futures Course. *Social Science Record*, 9(3).

Glenn, J. (Ed.). (1999). *Futures Research Methodology*. The United Nations University.

Grau, G., Franch, X., & Maiden, N. (2005). A Goal-Based Round-Trip Method for System Development. In *11th International Conference on Requirements Engineering: Foundations for Software Quality - REFSQ*.

Heckman, J., & Leamer, E. (Ed.). (2007). *Handbook of Econometrics (volume 6A)*. Amsterdam: Elsevier.

Javed, T., Maqsood, M., & Durrani, Q. (2004). A study to investigate the impact of requirements instability on software defects. *ACM SIGSOFT Software Engineering Notes*, 29(3), 1-7.

Jian, Y., Li, T., Liu, L., & Yu, E. (2010). Goal-Oriented Requirements Modelling for Running Systems. In *1st International Workshop on Requirements at Run-time - RRT*.

Jureta, I., Borgida, A., Ernst, N., & Mylopoulos, J. (2010). Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling. In *18th IEEE International Requirements Engineering Conference – RE* (pp. 115-124).

Kephart, J. (2005). Research challenges of autonomic computing. In *27th International Conference on Software Engineering – ICSE* (pp. 15-22).

Kephart, J., & Chess, D. (2003). The Vision of Autonomic Computing. *IEEE Computer*, 36(1), 41-50.

Kotonya, G., & Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. John Wiley & Sons.

Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In *5th IEEE International Symposium on Requirements Engineering* (pp. 249-262).

Lapouchnian, A., Yu, Y., Liaskos, S., & Mylopoulos, J. (2006). Requirements-Driven Design of Autonomic Application Software. In *16th Annual International Conference on Computer Science and Software Engineering - CASCON*.

Loveridge, D. (1996). Technology Foresight and Models of the Future. In *Policy Research in Engineering, Science and Technology Ideas in Progress*.

Lucena, M., Santos, E., Silva, M., Silva, C., Alencar, F., & Castro, J. (2008). Towards a Unified Metamodel for i*. In *2nd International Conference on Research Challenges in Information Science - RCIS* (pp. 237-246).

Mao, Y., Vassileva, J. & Grassmann, W. (2007). A System Dynamics Approach to Study Virtual Communities. In *40th Annual Hawaii International Conference on System Sciences – HICSS* (p. 178).

Morandini, M., Penserini, L., & Perini, A. (2008). Towards goal-oriented development of self-adaptive systems. In *Workshop on Software Engineering for Adaptive and Self-Managing Systems – SEAMS* (pp. 9–16).

Mylopoulos, J., Castro, J., & Kolp, M. (2000). Tropos: Toward agent-oriented information systems engineering. In *2nd International Bi-Conference Workshop on Agent-Oriented Information Systems - AOIS*.

Navarro, I., Leveson, N., & Lundqvist, K. (2000). *Reducing the Effects of Requirements Changes through System Design*. SERL report.

Neto, G., Gomes, A., Castro, J., & Sampaio, S. (2005). Integrating activity theory and organizational modeling for context of use analysis. In *Latin American conference on Human-computer interaction - CLIHC* (pp. 301-306).

Object Management Group – OMG. (2009). Business Process Model and Notation (BPMN) Version 1.2. Retrieved September 25, 2011, from http://www.omg.org/spec/BPMN/1.2/

Object Management Group – OMG. (2009). Unified Modeling Language. (UML) Version 2.2. Retrieved September 25, 2011, from http://www.omg.org/spec/UML/2.2/

Object Management Group – OMG. (2010). Object Constraint Language Version 2.2. Retrieved September 25, 2011, from http://www.omg.org/spec/OCL/2.2/

Oreizy, P., Gorlick, M., Taylor, R., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D., & Wolf, A. (1999). An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems and Their Applications*, 14(3), 54-62.

Oz, E. (1994). When Professional Standards are Lax, The CONFIRM failure and its Lessons. *Communications of the ACM*, 37(10), 29-43.

Pimentel, J., Franch, X. & Castro, J. (2011). Measuring Architectural Adaptability in *i** Models. In *XIV Ibero-American Conference on Software Engineering – CibSE* (pp. 115-128).

Pimentel, J., Lucena, M, Castro, J., Silva, C., Alencar, F., & Santos, E. (in press). Deriving Adaptable Software Architectures from Requirements Models: The STREAM-A Approach. *Requirements Engineering Journal*.

Pimentel, J., Santos, E., & Castro, J. (2010). Conditions for ignoring failures based on a requirements model. In *22nd International Conference on Software Engineering & Knowledge Engineering – SEKE* (pp. 48-53).

Pimentel, J., Castro, J., Perrelli, H., Santos, E., & Franch, X. (2011). Towards Anticipating Requirements Changes through Studies of the Future. In *5th International Conference on Research Challenges in Information Science - RCIS*.

Porter, A. et al. (2003). Technology Futures Analysis: Toward Integration of the Field and New Methods. *Technological Forecasting & Social Change*, 71, 287-303.

Qureshi, N., Perini, A., Ernst, N., & Mylopoulos, J. (2010). Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems. In *1st International Workshop on Requirements at Run-time - RRT*.

Rosenberg, L., & Hyatt, L. (1996). A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality. In *Product Assurance Symposium and Software Product Assurance Workshop* (p. 209).

Santos, E., Pimentel, J., Dermeval, D., Castro, J., & Pastor, O. (2011). Using NFR and Context to Deal with Adaptability in Business Process Models. In *2nd International Workshop on Requirements at Runtime - RRT*.

Schach, S. (2002). *Object-Oriented and Classical Software Engineering*. McGraw Hill.

Schwartz, P. (1991). *The Art of the Long View: Planning for the Future in an Uncertain World*. New York: Doubleday.

Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering: A good practice guide*, John Wiley & Sons.

Tesauro, G., & Kephart, J. (2000). Foresight-based pricing algorithms in agent economies. *Decision Support Systems*, 28(1-2), 49-60.

RAE, & BCS. (2004). *The Challenges of Complex IT Projects.* Royal Academy of Engineering and British Computer Society.

Tonn, B., Hemrick, A., & Conrad, F. (2006). Cognitive representations of the future: Survey results. *Futures*, 38(7), 810-829.

Wall, J., & Sinnadurai, N. (1998). The Past, present and future of EEE components for space application: COTS – The next generation. In *IEEE International Frequency Control Symposium* (pp. 392-404).

Yu, E., Giorgini, P., Maiden, N., & Mylopoulos, J. (Ed.). (2011). *Social Modeling for Requirements Engineering*. Cambridge, Massachusetts: The MIT Press.

Yu, Y., Leite, J., & Mylopoulos, J. (2004). From goals to aspects: discovering aspects from requirements goal models. In *12th IEEE International Requirements Engineering Conference - RE* (pp. 33-42).