Universidade Nova de Lisboa

Faculdade de Ciências e Tecnologia

Departamento de Informática

Dissertação de Mestrado em Engenharia Informática

1º Semestre, 2008/2009

Viewpoints and Goals: Towards an Integrated Approach

Manuel Filipe Pimenta

Nº 26224

Orientador

Prof. Doutor João Araújo Júnior

Lisboa

19 de Fevereiro de 2009

Nº do aluno: 26224

Nome: Manuel Filipe Prista Lucas Conrado Pimenta

Título da Dissertação:

Viewpoints and Goals: Towards an Integrated Approach

Palavras-Chave:

- Engenharia de requisitos orientada a pontos de vista
- Engenharia de requisitos orientada a objectivos
- PREview
- KAOS

Keywords:

- Viewpoint-oriented requirements engineering
- Goal-oriented requirements engineering
- PREview
- KAOS

Acknowledgments

There are several people I would like to acknowledge regarding this special chapter in my life.

Firstly I would like to thank my supervisor, Prof. Dr. João Araújo for the long hours he put in reading my never-ending texts and straining to cut their sizes in half. His help was essential to keep this document from a length of 500 pages or more, and the constant requests for deliverables kept me on track.

Secondly, to Ana Sofia Penim, who despite having to deliver her own dissertation had the generosity of always being there for me and for my short temper during stressful times. Also, I cannot forget the five years, going on six we spent working side by side in innumerous projects, my rambling and your organization. Thank you, you have ruined every possible team efforts for me in the future. I cannot thank you as a friend for you are much more than that.

Thirdly, I would like to thank my parents and closest family for all the support and understanding when all I could think about was finishing this work; especially my brother Nuno, mother Rosário and father António: thank you for understanding my brooding silences and my instantaneous ramblings.

Finally my closest friends: Tiago, my brother of choice; Miguel, my cousin, for always trying to drag me to soccer games and clearing my head; and Pablo for the constant nagging and concern.

I could not end without forwarding a special thank you to my grandfather. My teacher, my hero, my friend. Thank you for watching over me from up there.

Thank you all.

Resumo

A elicitação e a análise de requisitos têm sido estudadas de acordo com diversas abordagens que diferem maioritariamente na sua "orientação", neste caso baseando-se em *objectivos* ou *viewpoints*.

As abordagens orientadas a objectivos tal como KAOS baseiam-se em objectivos para dirigir o processo de elicitação de requisitos: um *objectivo* é uma meta que o sistema em consideração deve atingir e representa uma propriedade do mesmo que pode reflectir um requisito funcional (e.g. um serviço oferecido pelo sistema) ou não funcional (e.g. segurança, desempenho); a sua satisfação pode implicar a participação de vários agentes e a resolução de obstáculos que possam surgir. A abordagem KAOS oferece um método inequívoco para a decomposição dos requisitos e pode providenciar um conjunto de heurísticas a abordagens que não o tenham.

Abordagens orientadas a *viewpoints* tal como PREview focam a recolha de informação relativa ao problema a partir de vários agentes que podem ter perspectivas diferentes, igualmente válidas e incompletas sobre o problema. Estas perspectivas parciais reflectem as suas diferentes responsabilidades, papéis ou interpretações das fontes de informação; portanto a combinação do agente e da sua observação do sistema é chamada um *viewpoint*. PREview beneficia de uma abordagem particularmente leve à encapsulação de requisitos, mas falha em providenciar um conjunto de heurísticas para dirigir o seu processo de elicitação.

Considerando os factores identificados em cada abordagem, é de verificar que as abordagens são complementares: por um lado, KAOS oferece um conjunto de heurísticas para a elicitação de requisitos através da decomposição de *goals*; por outro lado, PREview é uma abordagem leve à engenharia de requisitos orientada a viewpoints dirigida especialmente à integração, faltando-lhe no entanto um mecanismo mais sistemático para dirigir o processo de elicitação dos requisitos.

O objectivo desta dissertação é então propôr uma abordagem híbrida que se baseia em PREview e aproveita os benefícios da abordagem KAOS. O resultado é sinergético no que diz respeito ao facto de, por exemplo, a completude ser melhor endereçada providenciando heurísticas de elicitação de requisitos.

Abstract

Requirements elicitation and analysis have been studied according to several approaches that differ mostly on their "orientation", in this case relying on *goals* or *viewpoints*.

Goal-Oriented approaches such as KAOS rely on goals to direct their process of eliciting requirements: a *goal* is an objective the system under consideration should achieve and represents a system property that may reflect either a functional (e.g. a service provided by the system) or a non-functional (e.g. security, performance) requirement; its satisfaction may imply the participation of several agents and the resolution of possible obstacles that may arise. The KAOS approach offers an unambiguous method for requirement decomposition and may provide a set of heuristics to approaches where one does not exist.

Viewpoint-Oriented approaches such as PREview focus on gathering information pertaining to the problem from several agents that may have different, often equally valid, and incomplete perspectives on the problem. These partial intakes reflect their different responsibilities, roles, goals, or interpretations of the information sources; hence the combination of the agent and its input on the system is called a *viewpoint*. PREview benefits from a particularly lightweight approach to requirements encapsulation, but fails to provide a set of heuristics for the process of identifying the system's requirements.

Considering the issues identified in each approach, it is verifiable that both approaches are complementary: on the one hand, KAOS offers a set of requirements elicitation heuristics through *goal* decomposition; on the other hand, PREview is a lightweight approach to viewpoint oriented requirements engineering, tailored especially for integration, however lacks a more systematic mechanism to guide the requirements elicitation process.

The objective of this dissertation is therefore to propose a hybrid approach that builds on the PREview approach and brings together the benefits of the KAOS approach. The result is synergetic where, for example, completion is better addressed by providing a set of heuristics for requirement elicitation.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Context and Problem Description

System requirements are specifications of what should be implemented regarding the services a system should provide and its operational constraints [23]. Requirements can be classified according to three types:

- *Functional* requirements are statements of services the system should provide [23] and describe how the system should react to particular situations;
- *Non-functional* requirements represent constraints on the system services, defining characteristics the system should reflect and overall qualities it should possess;
- *Domain* requirements come from the system environment and may be functional or non-functional.

Requirements Engineering (RE) entails all activities involved with the elicitation, analysis and management of all types of requirements in a systematic manner, in order to guarantee that requirements are complete, consistent and relevant.

These activities in RE have been studied according to several approaches that differ according to their drive and according to their perspective of the process itself. Although there are many techniques (e.g. Scenarios, Viewpoints, Goals), this work focuses on two of these fields of approaches: Goal-Oriented and Viewpoint-Oriented.

Goal-Oriented approaches, such as KAOS [13], rely on goals to direct their process of eliciting requirements, analyzing the system up for consideration regarding its several

organizational, operational and technical aspects and listing the detected problems and opportunities as goals to be achieved through the ensuing requirements.

A goal is an objective the system under consideration should achieve and represents a system property that may reflect either functional or non-functional requirements. By encompassing a set of requirements, a goal can be used for providing rationale for its "children", assessing their completeness and relevance, as well as their identification. Goal satisfaction may imply the participation of several agents and the resolution of possible obstacles that may arise.

The KAOS approach's modeling framework offers an unambiguous method for the identification of a system's requirements, the intervening agents and the relations between them, relying on the graphical representation of a requirements model [21]. This requirements model clearly identifies all the requirements in a project in a tree-like representation, detailing the intervening agents, related objects and even the operationalisation of the bottom-level requirements.

However, although KAOS is tailored for use with many differently sized projects, when working with large and complexly structured systems, a requirements document will most likely contain hundreds, if not thousands of requirements, and therefore, the correspondent KAOS models may prove to be too elaborate to be understandable by the system stakeholders. Seeing as one of the main goals of modeling requirements in such a way is to favor stakeholder comprehension of the proposed solution, a conflict of interests emerges. Hence, this approach would probably merit from some additional organizational elements.

Viewpoint-Oriented approaches, such as PREview [25], focus on gathering information pertaining to the problem from several agents that may have different, often equally valid, and incomplete perspectives on the problem [5]. These partial intakes reflect their different responsibilities, roles, goals, or interpretations of the information sources; hence the combination of the agent and its input on the system is called a viewpoint.

These approaches also focus on understanding and controlling the complexity of the systems under consideration by separating the interests of various actors, organizing them into hierarchies and formalizing this multi-perspective view into analysis methods. They strive to balance both the preservation of multiple perspectives during system development and the demands for consistency and coherence arising out of group work-focus [9]. This balance is also reflected when dealing with large projects since collecting system information from

several sources will inevitably result in a great deal of information to manage and a great deal of perspectives between which consistency must be assured.

However, the considered viewpoint-oriented approach (PREview) fails to provide a set of requirements elicitation heuristics, i.e. a mechanism to aid the analyst through the process, relying on his ability and instincts to understand the requirements that the elicited system concerns should entail and to assure the stakeholders that the requirement set is complete. Systematically deriving requirements from general goals could be a useful strategy for that purpose.

If one were to look at the issues identified in each approach, it is verifiable that both approaches are in some way complementary. On one hand, KAOS offers a set of requirements elicitation heuristics defined as goal decomposition techniques as well as mechanisms to verify the completeness of a given goal. However, it could probably benefit from a means to organize the elicited requirements in a stakeholder-friendly fashion. On the other hand, PREview focuses precisely on understanding and controlling the complexity of the systems under consideration by separating the interests of various actors and formalizing this multi-perspective view into analysis methods. However, it lacks a set of heuristics to guide the elicitation process, relying on the analyst's interpretation of the system's concerns for this task, which may lead to incomplete or ill-defined requirements.

1.2 Proposed Solution

Considering the enunciated problem description, the objective of this dissertation is therefore to propose a hybrid approach that brings together the advantages of both approaches and also solves some of the issues that were identified in each of them, particularly focusing on the PREview approach.

This hybrid approach would be based on the viewpoint oriented requirements engineering technique known as PREview and thus focus on addressing a system’s requirements elicitation efforts from an encapsulation perspective. It would incorporate the knowledge acquisition skills that KAOS provides for the requirements elicitation process and consequent

analysis, which would in turn contribute to mitigate PREview's previously referred weakness regarding the management of information and the elimination of redundant requirements, as well as the lack of heuristics for the elicitation process. The approach would also rely on the grouping mechanisms of PREview to bring a greater level of organization into the requirements model, since, in theory, complexity would be better addressed by organizing goals in terms of viewpoints.

Summing up, in this hybrid approach PREview would provide the basis to encapsulate requirements in a user-friendly fashion and address the issue of system complexity, while KAOS provides the requirements decomposition mechanisms required to derive them in a systematic way.

1.3 Main Contributions

As it has been explained thus far, the process of eliciting requirements and presenting them in such a way as to favor stakeholder comprehension has much to gain from the KAOS methodology. It is a clear and concise method for graphically representing the analyst's work and basically provides a map for understanding the way in which the system will achieve the proposed goals.

However, as it was previously explained, large KAOS models can be hard to comprehend, not because of fuzzy concept definition, but because of the quantity of information that might be present in a model at the same time. Furthermore, when defining the non-functional traits of the system, KAOS models can sometimes lack information that might be useful, namely when dealing with conflicts between these non-functional requirements.

On the other hand, although PREview viewpoints constitute useful grouping units regarding a set of requirements, the identification process of these same requirements lacks some form of established guidelines in order to provide insight into this particular step. It is to counteract these problems that this hybrid approach is proposed.

The main contributions of this dissertation are the following:

- A study into the several goal-oriented and viewpoint-oriented approaches is provided, establishing a state of the art and describing in some depth both the KAOS and PREview approaches as main representatives of each requirements elicitation strategy.
- A suitable case study is used to present some aspects of both the KAOS perspective and the PREview perspective in order to demonstrate each approach's capabilities and limitations regarding requirements elicitation.
- A study into each approach's conceptual-model will be provided, describing in what way a connection between the approaches could be established, namely relating the PREview viewpoints to the KAOS goals and requirements structuring.
- The approach will define how to use viewpoints to drive the segmentation of the KAOS requirements model into identifiable and easier to understand sections of the system, i.e. to reduce the complexity of the overall system requirements model.
- By associating system goals to system viewpoints, the requirements elicitation effort will be supported by the KAOS goal decomposition mechanism.

1.4 Document Structure

The present dissertation is organized as follows: a survey of the existing Viewpoint-Oriented approaches and the main concepts they entail is shown in **Chapter 2**; a survey of the existing Goal-Oriented methods and the main concepts involved is present in **Chapter 3**; the Hybrid approach is proposed and its heuristics are defined in **Chapter 4**, detailing the several steps based on a demonstrative case study; the application of the Hybrid Approach to a real world industry case study is detailed in **Chapter 5**, focusing on the advantages brought by the approach; conclusions are drawn in **Chapter 6**, along with references to future work developments.

Chapter 2
Related Work: Viewpoint-Oriented Requirements Engineering

This chapter outlines several of the Viewpoint-Oriented (VO) approaches currently existent or strongly referenced in current work. An introduction to VO methodology is presented, including definition of the generic concept of *viewpoint* and advantages in its use. Existing methods are then described and its current uses outlined, with a particular emphasis on the PREview methodology.

2.1 Main Concepts

Software development involves a complex combination of activities and generally entails an also complex description or model; knowledge of the application domain is essential, as well as experience in the several areas of software development process, methods, techniques and languages. In order to manage all these activities, one should be able to structure them so as to provide a sustainable organization for the software development process and its consequent specification [7].

Due to these knowledge demands, a software project normally involves many participants, with experts in various aspects of software development and in the application area; each of them being required to perform different roles, acquit certain responsibilities and input various concerns which may change as the software evolves. Also, due to the capture of

diverse perspectives from several stakeholders, conflicting opinions, possibilities of errors and omissions will arise [11].

A stakeholder can be a human, role, or organization with an interest in the system and this definition can also include both the customer's and developer's organizations. Developer's viewpoints may also include those of the analyst(s) and other disciplines involved in the system development [24].

In light of this fact, one can conclude that requirements elicitation from a single perspective inevitably focuses one stakeholder's needs at the expense of other viewpoints; similarly, attempting to synthesize multiple stakeholders into a singular perspective would be a very difficult task. Therefore, in order to reduce the risk of overlooking these several stakeholders and their requirements, it is essential to recognize "the distributed nature of the knowledge" they provide [7].

A viewpoint based approach to requirements analysis recognizes this distributed nature and thus provides a technique that adequately captures these different perspectives, their attributes and the relationships between them [11].

2.1.1 What is a viewpoint?

The concept of Viewpoint has evolved and diversified since its original explicit presentation in CORE [17], and even in SRD [18] and SADT [22], where the concept was present although without explicit reference. So, in order to define a viewpoint, one must take from the several proposed approaches their more generic descriptions.

Consider the example of a parking lot management system: what first comes to mind is to consider the user point of view since, at first glance, the only interaction that takes place is between a person entering the parking lot and the entry mechanism that regulates the cars that enter; however if one would elicit a set of requirements based solely on this point of view, one would get an incomplete list of functionalities. This can be explained simply by the fact that the parking lot system envelops not only the user point of view but also the entry-machine's point of view, the sensor that detects passing cars or even the requirements that come from the parking lot manager's responsibilities. As such, it follows that many times, more than one

point of view is required in order to encompass the whole scope of a system's functionalities, hence the idea of using viewpoints.

A Viewpoint provides a combination of an agent and its particular view on the system, "a standing or mental position used by an individual when examining or observing a universe of discourse" [14]. It captures the particular role of that agent and the several responsibilities offered by its different perspective, encapsulating only the aspects of the application relevant to that partition of the system's domain.

In software terms, a Viewpoint can be defined as an object, albeit a loosely coupled and locally managed one [8], which contains partial knowledge about the system and its domain, specified in a particular scheme or notation, and also partial knowledge of the design process.

Hence, a viewpoint may contain a set of requirements as well as a definition of the viewpoint's perspective, a list of the sources from which the requirements were elicited and a rationale for each requirement [24].

2.1.2 Why use viewpoints?

Many reasons can be supplied that justify the use of viewpoints, the most obvious one being the need to guarantee that no important requirements are overlooked during the elicitation stage; by using viewpoints and allocating time to their explicit identification, it is more likely that a greater number of stakeholders are recognized and consequently their requirements.

Regarding expertise and knowledge of the domain, using viewpoints allows for the engineers developing the system that may have experience with similar systems, to suggest requirements derived from that experience. Also, seeing as system maintenance represents a large percentage of the development costs, by collecting contributions from the technical staff that will have to manage and maintain the system, one can simplify system support and therefore greatly reduce the budget allotted for system maintenance.

Viewpoints also provide a clear separation of concerns by permitting the development of a set of partial specifications in isolation from other viewpoints, avoiding conflicts during the elicitation stage and allowing for better informed trade-offs between requirements when necessary.

Also, in latter stages of an application's development, there may come a time when requirements need to be traced back to their sources for further clarification, by associating them to separate viewpoints this traceability is greatly enhanced and a task that might have required a great deal of backtracking is reduced to a simple consulting of the Requirements Document [24, 8].

Furthermore, viewpoints are complementary to a number of other requirements engineering practices. In particular, they can help inform the development of system models, seeing as they can be derived from each viewpoint in order to clarify their requirements.

They are useful to partition the domain information and allow for simpler formal representations of software specifications, which are in turn described as configurations of related Viewpoints; this facilitates distributed development, the use of multiple representation schemes and scalability.

Seeing as the several roles may be organizationally defined, Viewpoints can also help prioritize and manage requirements according to some hierarchy or even be used as a way of classifying stakeholders and other sources of requirements according to three generic types [23]:

- **Interactor viewpoints** represent people or other systems that interact directly with the system and are, many times, the first to be elicited since they are the most obvious ones;
- **Indirect viewpoints** represent stakeholders who do not use the system themselves but who influence the requirements in some way through their claim in the application's development;
- **Domain viewpoints** represent domain characteristics and constraints that influence the system requirements and most often generate the non-functional system requirements.

As one can see, Viewpoints are not only concerned with human perspectives; beyond being associated with the system's stakeholders, viewpoints may also refer to the system's operating environment or even other systems that will, in some way, interact with the application being developed.

Therefore, Stakeholders are not the only sources of requirements since the system is always installed in an environment that includes other systems or components with which it must

exchange data or control and these exchanges impose requirements. For instance, an interfaced system or component has a perspective on the system just as the stakeholder does; hence it makes sense to use a viewpoint to model that particular perspective [24].

In addition, many applications are subject to the influence of the domain that concerns them; in some, the domain-imposed requirements are trivially obvious, constituting explicit concerns from the application environment; in others, they are implicit in the principal stakeholders' requirements and may require a careful analysis of their responsibilities or stakes in the application [24]. They may come from several departments in an organization and many times concern both a favorable image for the organization and product functionality for the user. Nonetheless, domain derived requirements are extremely important to contemplate, since they represent the main factors that differentiate between projects, and they are also the most difficult to appease.

2.1.3 Summary

Viewpoint-based approaches offer a clear understanding of the "distributed nature" of many systems' requirement elicitation by offering a separate but structured way of acknowledging the several stakeholders and their responsibilities in the project being developed. Viewpoints provide a framework for organizing these requirements and improve traceability and scalability in larger applications where following a spiral model, that many times generate constant changes, is required.

In familiar domains where systems are highly constrained by inherent or explicit concerns, viewpoints probably do not offer significant advantages, also with developing systems where there are few or homogeneous sources of requirements, viewpoints may offer little advantage. However, in systems where requirements rework is frequently performed or where there are problems with managing requirements from different sources, then viewpoints may offer significant advantages and provide the "edge" that one needs to make it all work [24].

2.2 Existing Methodologies

Since CORE [17], where the use of viewpoints was first made explicit, several requirement definition methodologies have surfaced that are best suited to different activities in the software process. The most relevant methods proposed so far are presented in the following sections.

2.2.1 SRD

The Structured Requirements Definition (SRD) methodology [18] focuses on the structuring of input and output data, which assists the analyst in identifying key information objects and operations on those objects. It stems its name from a book by Ken Orr from 1981, where he gives the basis for what he calls "defining the application context", and details it in a four step process:

1. Construct a user level data-flow diagram (DFD) for each interview performed on organization individuals that perform some form of task relevant to the system, recording the input and output data obtained in the model;

2. Combine all user-level DFDs in order to create an integrated DFD. By compiling all relevant input/output data, it is possible to resolve conflicting data flows at this stage through a careful analysis of the designed model;

3. Construct the application-level DFD by recognizing the part of the organization being analyzed in the combined user-level DFD and underlining it by drawing a dotted line around it;

4. Define the application-level functionalities by listing and relating each activity being performed to its corresponding function in the system.

The SRD perspectives/viewpoints relate to individuals that perform some manual task that requires automation, therefore requirement elicitation can only be done by interviewing those individuals. In highly interactive activities this proves to be a valid approach, seeing as most of the system's requirements are human-derived ones; however, in activities that envelop little

human interaction, like the example of the parking lot management system, it is almost impossible to obtain the whole scope of the application from the interview of one individual.

Therefore, information that could be obtained from several other systems or even individuals that simply cannot be interviewed is wasted, when it could contribute to the overall understanding of the system since they represent external viewpoints. Thus, SRD presents an intuitive, rather than a defined notion of a viewpoint, and "does not work well if the system being specified does not involve the automation of several manual tasks" [11].

2.2.2 SADT

SADT (Structured Analysis and Design Technique) [22] is a superset of Structured Analysis and was the first graphically oriented method developed for use in performing requirements definition. It was developed in the early 1970s by Ross at SoftTech, initially as a method to provide architecture documentation for large and complex systems, and later as a methodology to deal with system development complexity through a team-oriented approach with textual and graphical support [22, 3].

SADT consists of two basic components: the diagramming language of structured analysis (SA) and the design technique (DT) that offers the discipline of thought and action that is needed to use the SA language. The diagramming language is based on a data-flow model that represents the system's more relevant activities and provides the basis for functional decomposition: a rectangular box representing the system's "most abstract activity", a set of data input, data output and control arrows. Consistency maintenance can be done by checking each level of functionalities with its higher functions, through comparison of their respective input and output information [22, 11].

The SADT viewpoint, however, is not defined; it is derived intuitively from the modeling technique and is not analyzed beyond being seen as a data source or sink. Also, SADT does not provide a framework for associating control information with viewpoints on a broader scale [11].

2.2.3 CORE

CORE (COntrolled Requirement Expression) [17] is a "functional requirements definition method" developed for British Aerospace (BA) in the late 70s by System Designers, defined to fit into a set of standards and procedures produced by the same company. Since then, it has been used in several aerospace and defense applications in the United Kingdom (the more notable ones being the EAP in the 80s and the EFA in the 90s) [11].

It is based on the idea of partitioning a system according to the notion of viewpoints, each with an associated client authority [9], having been the first method to explicitly define and adopt them as an element in requirements structuring. CORE defines its viewpoints according to their functional or non-functional aspect on a first level and as bounding or defining, concerning their direct or indirect interaction with the system.

The CORE methodology defines the steps in production of a requirement specification, with particular emphasis on start-up and the link between steps. These steps comprise viewpoint identification and structuring, tabular collection, data structuring, single and combined viewpoint modeling and constraint analysis.

This step-by-step process starts off by identifying possible viewpoints and classifying them as functional or non-functional through means of a brainstorming session composed of the users, buyers and system developers. The final step in viewpoint identification involves dividing functional viewpoints into a set of bounding and defining viewpoints.

Viewpoint structuring provides a framework for the capture and consequent analysis of the system's requirements, iteratively decomposing it into a hierarchy of functional sub-systems, each of them constituting a viewpoint, and placing structurally bounding viewpoints at the same level as the target system. This hierarchy obviously separates the tasks to be analyzed into smaller specification levels, allowing for the amount of detail at each level to be controlled and its analysis simplified.

The use of tabular collection as seen in Figure 1-1 provides a mechanism for collecting and organizing information about a viewpoint with respect to the action it performs, the input data used for these actions, the output data derived and the sources and destinations of the data. Through this, CORE is able to resolve omissions and conflicts related to the exchange of

information across viewpoints by checking that outputs from one tabular collection correspond to input data in their destination viewpoints.

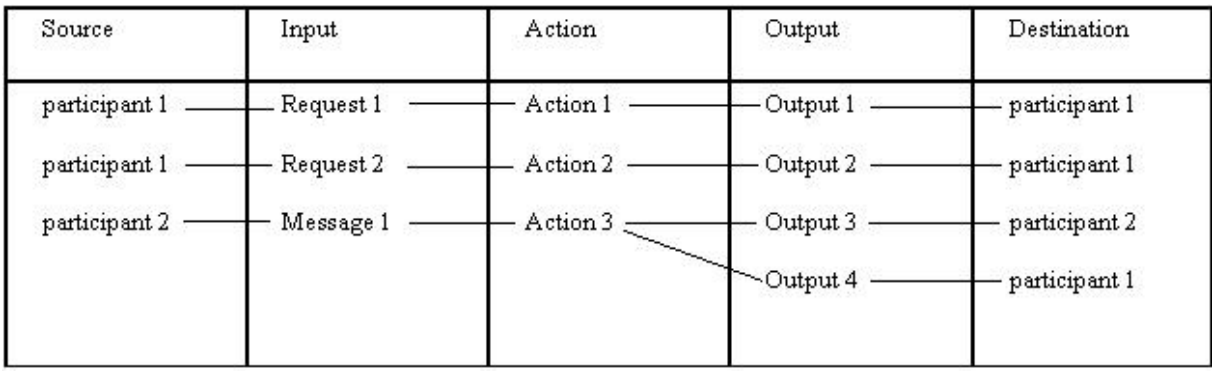

Figure 1-1 Tabular Collection

However, CORE presents some weaknesses. The lack of a properly defined notion of a viewpoint, seeing as it can be any entity that exchanges information with the system, makes it a difficult task to say what constitutes a valid viewpoint and what does not. When one considers the distinction between bounding and defining viewpoints, this just contributes to the problem by requiring the differentiation of external entities and sub-processes. Furthermore, despite the previously mentioned guidelines, CORE also acknowledges that two analysts specifying the same problem are likely to come up with different solutions and viewpoints.

Also, CORE concentrates its efforts on analyzing internal (defining) viewpoints. Bounding viewpoints, which represent the system's interaction environment and so being of special interest, are not analyzed beyond being seen as data input or output [11, 17].

2.2.4 VOSD

In Viewpoints-Oriented Software Development (VOSD) [7], Finkelstein argues that developing large software systems involves the participation of many experts, in various aspects of the software development and the application area. Furthermore, each of these participants constitute a constantly changing and evolving set of responsibilities and concerns

that follow the software's development, as well as particular knowledge that may provide further insight into the specifications being developed. He further contends that the solution for the management of these distributed activities and sources of knowledge is to provide a structurally sound partitioning method that can accompany the software development process and the subsequent specification.

Finkelstein regards viewpoints as a way to capture the role attributed to a participant and the responsibilities it ensues at each stage of the development process. A VOSD viewpoint is defined as a "loosely coupled, locally managed object which encapsulates partial knowledge about the application domain, specified in a particular suitable formal representation, and partial knowledge of the process of software development". It includes information about the domain it relates to, a specification, a work plan that establishes the constraints that the specification must obey and a work record. A viewpoint with information only regarding a style and a work plan is called a template.

The concept of allowing for different representation schemes is an important one in VOSD and much emphasis is placed on the designing of templates that may offer some measure of standardization while allowing for each member of the development team to use the style he or she is most comfortable with or that is best suited for the sub-problem at hand. Viewpoints can therefore be organized in configurations that establish some form of relationships between them.

In a final analysis, Finkelstein fails to provide a firm framework for resolving the conflicts between representation schemes with little correspondence and therefore restricts the development theme to a careful selection of related schemes; for instance, establishing an obvious relationship between object-oriented models and data-flow models may require substantial refactoring in order to be useful. It should also be mentioned that VOSD provides no obvious way of integrating functional and non-functional requirements and capturing the different classes of interacting entities [11, 7].

2.2.5 VA

In [14], Leite studies viewpoint resolution as a means for very early validation in the process of requirements elicitation. By relying on the principle that a greater number of sources of

information guarantee a better understanding of the overall problem, one must also be able to guarantee their validity as sources for requirement collecting. Therefore, Leite focuses on the very early validation of viewpoints as a key for the identification and classification of problems related to completeness and correctness.

Leite presents four essential definitions in his proposal:

- A **Viewpoint** is a standing or mental position used by an individual when examining a universe of discourse (the overall context in which the software will be developed, including all the sources of information and all the people related to the project);
- A **Perspective** is a set of facts observed and modeled according to a particular aspect of reality. He considers as modeling aspects: the data perspective, the actor perspective and the process perspective;
- A **View** is defined as an integration of these perspectives, which is achieved by a view-construction process;
- A **Hierarchy** may be defined as a "is-a" hierarchy or a "parts-of" hierarchy of concepts in the universe of discourse.

His approach comprises procedures to formalize viewpoints (method), to analyze the formalized viewpoints (static analyzer) and a viewpoint language (VWPL) to represent them. The method consists of at least two analysts (viewpoints) describing their intake on the universe of discourse using VWPL to express it. Each of them analyses the data, process and actor perspectives, and the hierarchies to improve their own perception, solves the internal conflicts that might arise and integrates the final perception into a view, which is constructed by means of the three perspectives and two hierarchies. After that, both viewpoints are compared and analyzed and a list of discrepancies between perspectives and hierarchies and their types is produced.

The final step is the integration of the perspectives into a view and after two of them are available, it is possible to compare different viewpoints for correctness and completeness [14, 11].

2.2.6 VOSE

The VOSE (Viewpoint-Oriented Software Engineering) framework was proposed by Finkelstein et al. [8] and supports the use of multiple perspectives in system development, providing a means for developing and applying methods for system design. It relies on viewpoints to partition the system specification, the development method and the formal representations in which the system is specified.

VOSE starts off by identifying the problems in organizing development in a setting with many actors, diverse domain knowledge and development strategies, but especially several representation schemes. This framework then focuses on the development of heterogeneous and composite systems, in which these settings occur, by allowing the use of these different representation schemes in a structured, organized and recognizable framework, using viewpoints as the basic unit.

A VOSE viewpoint is seen as a building block for the framework; it is thought of as the combination of an actor (a participant) and his role in the development process, including the knowledge he brings to the project. In order to allow heterogeneous representation schemes, VOSE viewpoints have a specific composition:

- A representation **style**, in a notation that is particular to each developer's area or line of work;
- A **domain** that outlines the world in which the scheme is developed;
- A **specification** that describes particular domains in the style chosen;
- A **work plan** that describes the specification process;
- A **work record** that registers the history and state of development, detailing performed actions.

A work plan is the most complex part of the VOSE viewpoint and is divided into four types of actions: assembly actions are the ones available to the developer to build a specification (add, remove, etc); check actions concern consistency checks done to the specification, be them in-viewpoint that resolve simple inconsistencies, or inter-viewpoint checks that must either be "fought out" between developers, or transferred from a specification to another according to a

hierarchic structure; viewpoint actions create new viewpoints according to pre-defined templates; and guide actions establish a type of schedule for the developer to follow.

A viewpoint template is also one of the most important VOSE concepts to be introduced. Since that the allowance for different representation schemes may originate truly heterogeneous solutions, the existence of an organizational "standard" is essential to, as previously stated, guarantee some measure of order and consistency maintenance. So, as development proceeds and new viewpoints might be required, a viewpoint template as seen in Figure 1-2 may be created containing the specification, domain and work record slots empty, and that afterwards may be instantiated in different ways.

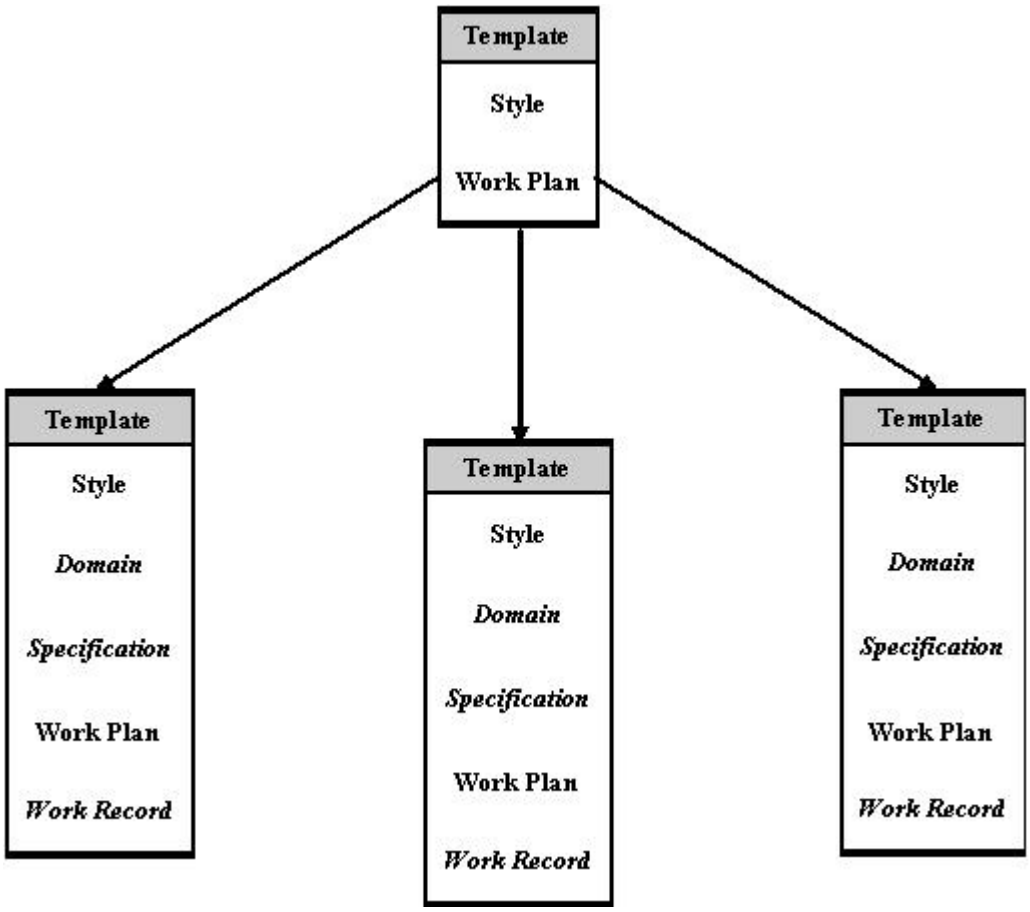

Figure 1-2 VOSE Templates

Also worthy of reference is the fact that the VOSE framework may not only be used from the method user's perspective, but also from the method developer's perspective; the framework

provides means to describe methods like SADT or CORE as configurations of viewpoint templates.

However, some of the advantages that VOSE presents come with clear drawbacks. Its support for managing inconsistency relies on the assumption that the analysts have the ability to write inter-viewpoint consistency rules, also, the automation of these rules' analysis would require them to be written in such a way that allowed for clear and unambiguous understanding (formal notation), which in itself may not suffice to explain all the relationships between requirements.

Another drawback is VOSE's demands of a formal description of each viewpoint, i.e. although it may tolerate some inconsistency, it does not allow for informal process description nor does it allocate space in a viewpoint's description for an informal explanation.

These drawbacks clearly limit the application of VOSE in most projects seeing as they imply a high introductory cost and a cost-ineffective management of constantly evolving requirements [25].

2.2.7 VOA

Viewpoint-Oriented Analysis (VOA) is an object-oriented viewpoint-based approach to requirements definition proposed by Kotonya and Sommerville (K\&S) [11]. It extends the definitions of viewpoints existent until then from its input/output orientations to incorporate viewpoint classification and different aspects of interaction. VOA also supports requirements capture and resolution, classification and analysis of viewpoint-system interaction and integration of functional and non-functional aspects of requirements. It is service-oriented, since it concerns interaction between entities that may exist in the surrounding system environments and in the system itself.

Regarding system viewpoints, VOA defines viewpoint identification, structuring and decomposition, information collection and reconciliation of information across viewpoints as its four main stages. K\&S further state that because of this staging, viewpoints can be used to formulate and structure requirements in a way that lends visibility to the deeper interaction between the system and its environment.

Viewpoint identifying normally places most of its reliance on stakeholders (in VOA identified as 'system authorities') so, in an attempt to improve previous viewpoint definitions, VOA adopts a pragmatic one that is both unambiguous and complete: a viewpoint is an external entity that interacts with the system, but does not require its existence in order to define itself. This means that it can interact with the system by receiving one or more services (functional requirements), providing control information to the system in order to start or stop service provision and, finally, providing the system with the data required for service provision.

This notion of viewpoint sheds the previous mixed definitions of both internal and external viewpoints in an attempt to simplify the process of requirements formulation; it does this by enabling a direct analysis, capturing the natural interaction between system and environment, distinguishing between types of viewpoint interaction and providing a framework for associating services with their constraints and enabling their reutilization.

VOA also separates viewpoints according to their ability to initiate interaction with the system; active viewpoints normally correspond to the customer's perspective, seeing as they provide control information **to** the system; passive viewpoints are essentially representative of data stores or sinks and control is directed **by** the system.

As stated before, the VOA viewpoint definition relies heavily on the integration between functional and non-functional requirements. It acknowledges that in every application, parallel to functional requirements (services), exist the non-functional ones that define the overall qualities or attributes of the resulting solution, called constraints. Each service provided to a viewpoint is therefore associated with the application's constraints (timing, reliability, system costs, etc) in a way that previously did not exist. Furthermore, recognizing that integrating functional and non-functional requirements along different viewpoints may generate conflicts and constraint overlapping, VOA suggests the early validation of these constraints by comparison of all provisions of each service across viewpoints. By identifying and solving these conflicts at an early stage, it aims for easier conciliation of clashes and inconsistencies.

Control information management is addressed by VOA from both the viewpoint and the system's perspective. A viewpoint directs control information by the description of signals and stimuli that start and stop the provision of services; the system provides viewpoints with entities (normally functions) that will relate to them in matters of provision of services.

As with CORE, VOA enables viewpoint structuring and decomposition by splitting the analytical tasks associated with viewpoints into a number of specification levels, diminishing the amount of detail that must be analyzed at each level. However, VOA extends the notion of hierarchy to contemplate in-viewpoint levels of abstraction, the top levels being the most abstract. This works in a similar fashion as class inheritance: all sub-levels inherit the services, attributes, control information and constraints of their top levels, being able to specify their own constraints on inherited services and to specify their own service requirements.

Information collection in VOA is handled by means of a form-oriented approach: for a viewpoint a form collects information regarding services specification, control information and data; for a service a form collects information on its constraints. This approach constitutes the basis for checking the completeness and correctness of information across viewpoints and for early constraint validation regarding services provided, thus enabling the information reconciliation stage, and final stage, referred in the beginning.

2.2.8 VORD

Viewpoint Oriented Requirements Definition (VORD) [12] is a method for requirements engineering that covers its process from initial requirements discovery to system modeling. Its authors, having been responsible for the VOA method, recognized that it tended to focus the RE process on the user issues rather than organization concerns, thus leading to incomplete system requirements. In order to encompass organizational requirements and concerns, they extended the concept of viewpoints to consider other inputs apart from direct clients of the system.

As the method it extends, VORD addresses firstly the issues of viewpoint identification, structuring, documentation, requirements analysis and specification. In order to do this, VORD relies on a notion of viewpoint based on the entities whose requirements are responsible for, or may constrain, the development of the intended system, and so they are also called *requirements sources*. Each of them relates to the system according to its needs and the interaction between them, and they fall into two major classes: *direct* viewpoints correspond directly to clients in that they exchange control information and services with the

system; *indirect* viewpoints have an interest in some of the services the system provides but do not interact directly with it.

VORD further extends the VOA treatment of viewpoints by generalizing the previous notion of 'system authority' into a set of viewpoint classes organized in a hierarchic structure. There is no generic structure, since each organization must establish its own hierarchy of viewpoint classes based on its needs and the application domain in which the system will exist, then constituting an important resource for the organization.

Viewpoint identification begins with abstract statements of organizational requirements and abstract viewpoint classes; VORD then proceeds with a pruning of the abstract class hierarchy that eliminates irrelevant viewpoint classes and follows by identifying stakeholder representative classes that are not present, sub-systems, system operators and individuals associated with indirect viewpoint classes; for all of these there may be associated viewpoints that must be contemplated in the class hierarchy.

The documentation of viewpoints in VORD is addressed with special attention; seeing as viewpoints are associated with a set of requirements, sources and constraints, and that each requirement is made up of a set or services, non-functional constraints and control requirements, VORD proposes that each viewpoint have an associated template, providing a structure for documenting detailed viewpoint requirements and events depicting its interaction with the proposed system. This offers a framework for formulating very detailed specifications, yet maintaining a clear separation of concerns.

Tracing the exchange of control information was also one of the main issues addressed by VORD. It was necessary to have a simple mechanism to address control requirements from a user's perspective, trace system-level control to viewpoints, expose conflict between control requirements and capture the distributed nature of control. VORD proposes the use of event scenarios, a sequence of events and exceptions that may arise during the exchange of information between the viewpoint and the system.

Event scenarios describe, at a top level, the normal course of events that a specific task might follow, what VORD calls a normal scenario, and as exceptions may occur to the task in question, event scenarios contemplate a branching of the main path in a series of layered sequence of possible events.

Viewpoint services specification aims to allow for multiple representations, seeing as a template is accessible to several levels of capabilities; non-technical staff may have to read it and understand it, as well as the need for a formal specification must be faced in order to comply with correctness demands. Therefore, VORD aims for a possibly ambiguous approach to specification, but one that allows for greater readability, which in many cases compensates the previous.

Last but not least, viewpoint analysis is intended to guarantee viewpoint correctness and completeness, but it is a difficult subject for developers to face. For completeness, VORD uses the previously referred template structure to guarantee a maximum of stored data and traceability allowance and this fact also helps with the correctness analysis. VORD incorporates a mechanism for attributing weights to non-functional requirements and comparing them and their use across the several viewpoints; this way, and making use of the traceability advantage, VORD allows for an easier and faster conflict resolution.

2.3 PREview

This particular approach merits a more profound analysis since it will be used in further work. Therefore, the following sections will present a careful description of its process and several stages, finalizing with a summary of the approach and a justification for its choice as basis for future work.

Furthermore, this approach is illustrated by a case study relating the Via Verde system, providing examples along the PREview process of a possible application of this method.

2.3.1 The PREview Method

The PREview (Process and Requirements Engineering Viewpoints) method [25] was proposed by Sommerville et al. in the mid 1990s in an attempt to handle what they called the "messy reality of requirements engineering". According to the authors there was a serious need for improving the quality of requirements specification, and that could be achieved in two ways: by improving the requirements engineering process so as to be less error-prone and

by improving the manner in which the specification itself was organized and specified so as to be more validation-friendly.

The use of viewpoints had already tried to address both of these improvement dimensions by collecting and analyzing requirements from multiple perspectives and providing a structure for the requirement elicitation and specification processes. However, the authors state that previous unstructured methods simply could not provide a reasonable support for viewpoint oriented requirements handling, and the ones that provided some structure were simply too rigid.

Also to be considered is the fact that until that time, only CORE had made the leap from theoretical proposal into industrial use, and even so, it was restricted to UK defense contractors. This situation could be explained by several factors, namely the overall inflexibility of the then current viewpoint models, the demand for fixed notations when defining requirements, limited support for both requirement evolution and negotiation, the non-existence of industrial tools that met the support requirements that the then current approaches required, a lack of recognition of the importance of non-functional requirements and also the incompatibility of the approaches with other methods that companies might have already implemented.

It is clear then that although viewpoint-oriented approaches constituted at the time, and still do, an interesting solution for most of the requirements engineering problems, there were also some issues with the introduction of viewpoints into the RE processes; there was a need for a lighter approach that could be introduced at relatively low cost and required an evolutionary, instead of revolutionary, process improvement.

PREview was originally designed to improve the processes of requirements discovery, analysis and negotiation, separating its scope from previous proposals by restricting itself to requirements elicitation. It possesses several key characteristics:

- Since requirements may be elicited from different sources with their particular notations, PREview allows for requirements that are associated with a viewpoint to be expressed in any manner. Although natural language and diagrams are the most common, it also admits for structured or formal notations.

- Non-functional requirements, or *concerns*, drive the analytical effort since they represent inherent characteristics of the system. They can be seen as generalizations of

the notion of goal, including both organizational goals and constraints that restrict the system.

- Viewpoints as basic building blocks possess a limited and explicitly described perspective in order to facilitate their requirements' discovery and analysis.

Because of all of these factors, PREview constitutes an approach that offers the required flexibility and allows users to define viewpoints appropriate to their application, complementing the standard notion of viewpoint with that of an organizational concern and following the more accurate spiral model of the requirements engineering process, as shown in Figure 1-3.

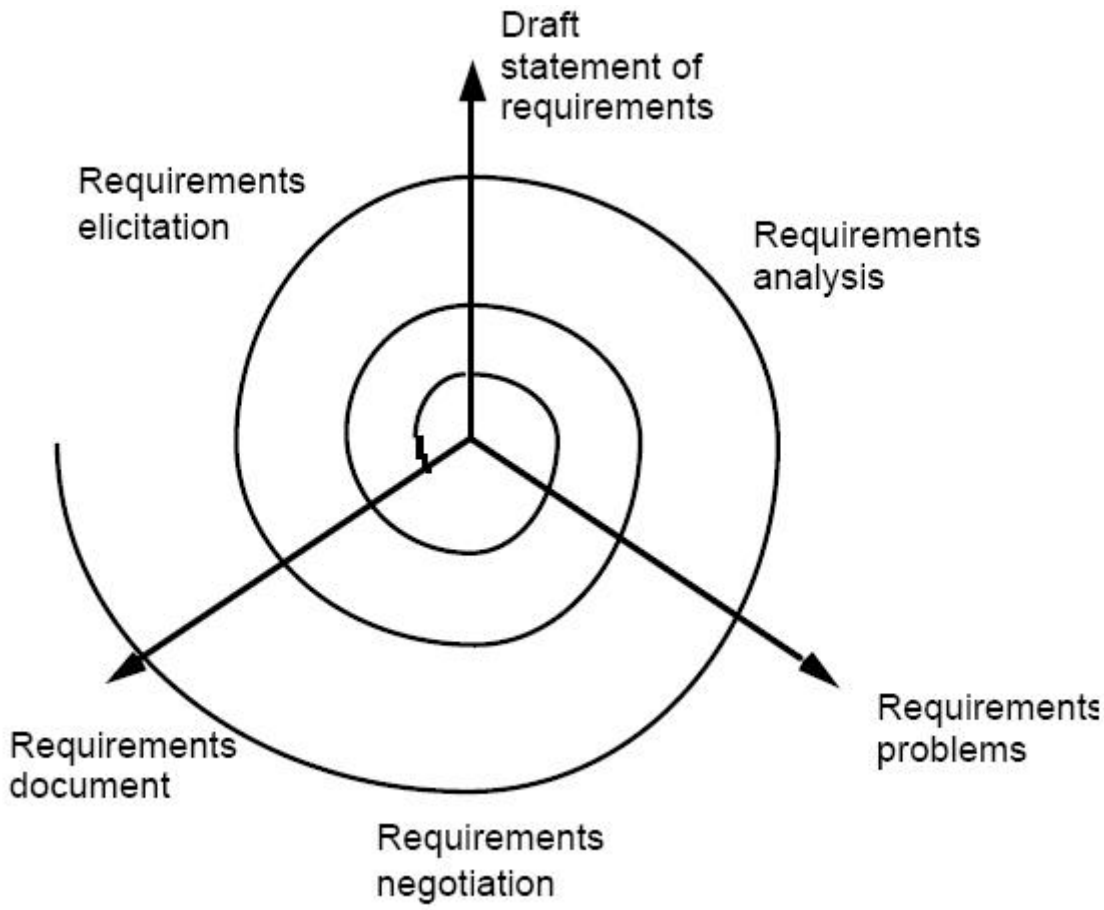

Figure 1-3 The Spiral Model of the Requirements Engineering Process

2.3.2 The PREview Viewpoint

As was the case in several previous methods, a PREview viewpoint is an encapsulation unit, a means to contain some but not all information about a system's requirements that may come from analysis of existing processes, discussions with the stakeholders or simply from domain and organizational information. A complete view of the system can thus be obtained by integrating requirements derived from different viewpoints.

A PREview viewpoint consists of the following information:

- **Name:** it is used to identify the Viewpoint and should be chosen in a way that reflects the Viewpoint's focus. It can also reflect a stakeholder's role in an organization or a part of the major system or even a process for which the analysis is restricted.
- **Focus:** it defines the Viewpoint and is attributed according to the viewpoint's perspective. Some emphasis is put into defining what is the *focus* of a viewpoint and is explained later.
- **Concerns:** as a default it includes all of the system's concerns, reflecting their orthogonal nature; however, some may be eliminated if it is proven that they have no relevance for a specific Viewpoint.
- **Sources:** it represents the explicit identification of the information sources associated with the viewpoint.
- **Requirements:** this is the set of requirements that one may find through system analysis according to the viewpoint's perspective and by consulting with the sources.
- **History:** this records changes in the viewpoint and assists with its evolution. It includes focus, sources and requirement changes.

2.3.3 Viewpoint Concerns

The definition of *concerns* was introduced in PREview as a way to establish explicit links between requirements and organizational goals or priorities, since they are defined as "high-level strategic objectives for the system" [25]. They provide a method to ensure that elicited

requirements do not conflict with goals or characteristics specific to the procuring organization.

Concerns may relate to several aspects of a system like *safety* or *functionality* and are established as a result of several discussions between the system's stakeholders, especially those in charge of strategic management. They are initially defined at a very high level of abstraction and must not amount to more than 6 or 7, due to their orthogonal nature, taking in account that the rigorous filtering process should only output the "most overriding, system-wide high-level goals and constraints" [25].

Another fact to consider is the distinction between a concern and a viewpoint, seeing as some types of concerns may closely resemble a type of viewpoint. Regarding this issue there are several key distinctions:

- Concerns cut across *all* viewpoints in a sense that the questions generated by studying each concern must be posed to *all* viewpoint sources, thus allowing for a better understanding of their effect in each of them.
- Concerns represent critical aspects of the system in development, organizational properties and are in fact what drive the requirement analysis process.
- Concern "requirements" apply to the system as a whole, across module boundaries, are inherited by all viewpoints, and viewpoint requirements must not conflict with them.

Finally, concerns are also subject to decomposition into levels of details regarding subsets of the problem space. Each of them translates into questions that serve as a checklist for checking against viewpoint requirements and verify if there are constraint infringements. Another usefulness of detailing concerns is the fact that perhaps their orthogonal nature is sometimes unclear to some analysts or even stakeholders; therefore, by separating a concern into "areas of interest", it is easier to comprehend the scope of their effect on the system.

2.3.4 Case Study: The "Via Verde" System

This case study consists of a system for automatic management of vehicle entrances and exits from adherent parking lots. The system makes use of an identifying gizmo that all cars must

possess and that can be recognized by the specific machines that exist in parking lot entrances and exits.

In order to obtain a gizmo a user must supply his personal information and afterwards must validate the same gizmo by associating it to a bank account in an ATM.

To access a parking lot, a user must place the gizmo in a readable place (most commonly the vehicle's windshield) and approach one of the entry machines; the machine recognizes the gizmo and, if it is a valid one, turns on a green light and allows the vehicle to enter. To exit the parking lot the gizmo recognition process is the same, adding to it the calculus of the amount due to the user's stay in the parking lot, which is shown in a display unit on the exit machine and added to the user's weekly total. This weekly total is sent to the bank in order for it to be charged to the user's account and in case there is some problem with the transaction, the bank should notify the system and the Accounting Department (AD) should send a letter to the respective client.

In case there are any issues with either the entrances or exits, a user may approach a parking lot employee in order to register a complaint in the system.

Finally, the system also contemplates informing the user of the gizmo's utilization by sending a monthly status report.

2.3.5 The PREview Process

As was previously stated, PREview concerns itself with activities related to requirements elicitation and follows a process as depicted in Figure 1-4, fitting into the axes of the spiral model of the requirements engineering process. As such, it is divided in three stages of a cycle: requirements discovery, requirements analysis and requirements negotiation. This cycle is normally iterated until the set of viewpoints, concerns and their respective requirements achieve a stable equilibrium and can finally be integrated into a requirements document.

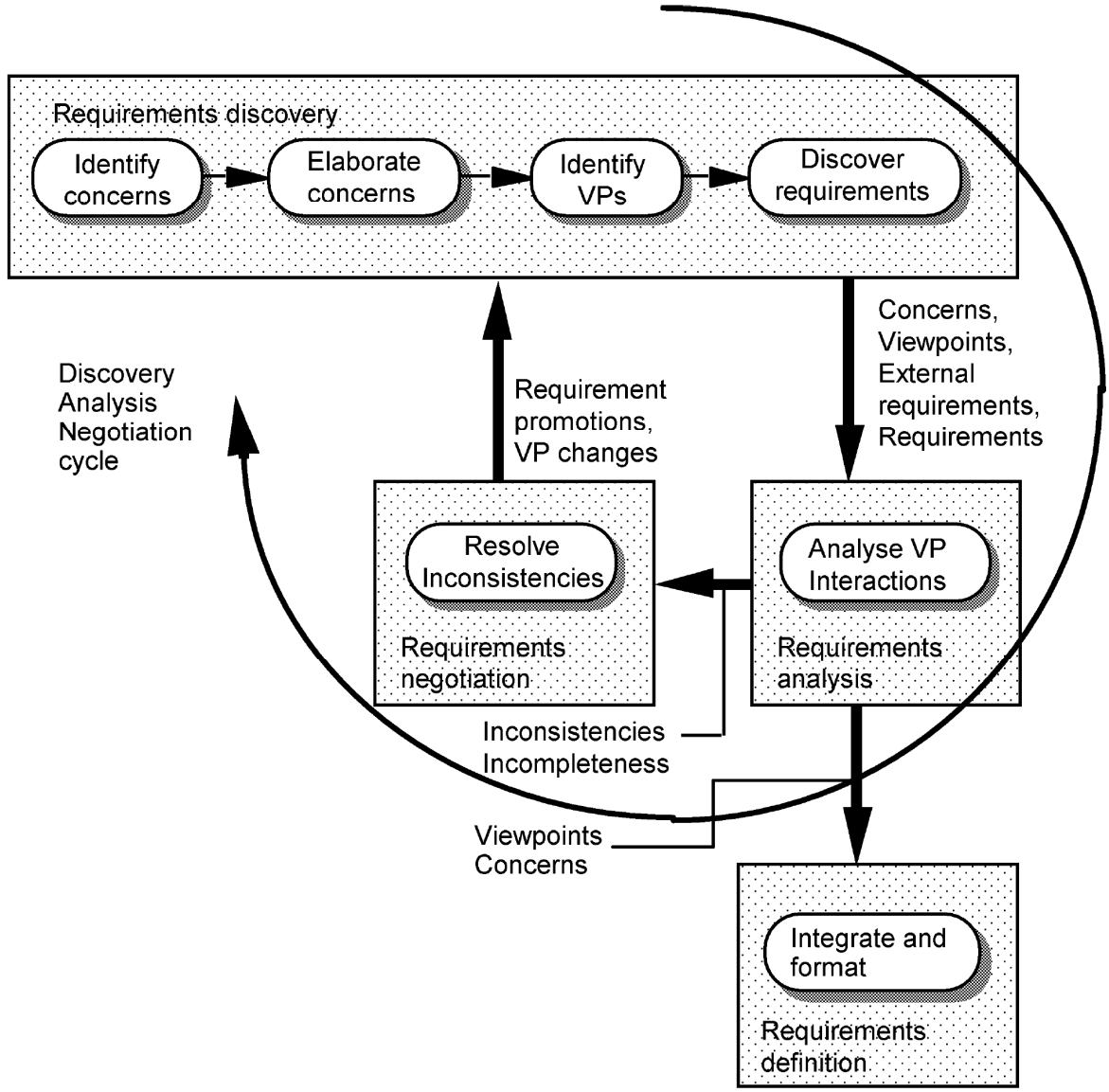

Figure 1-4 The PREview Process

2.3.5.1 Requirements Discovery

In PREview, the stage of requirements discovery is broken down into a succession of several activities:

1. Identification of concerns

2. Elaboration of concerns into requirements and questions
3. Identification of viewpoints
4. Discovery of each viewpoint's requirements

One of the defining traits of this approach is the importance of concerns in the entire process; they provide a transversal understanding of the several aspects of the system and the environment where it will be inserted, driving the following viewpoint and requirements discovery process and concentrating it on the essential factors of the system.

As is stated in [25], the first step in order to guarantee their proper identification is for an analyst to ask himself "what fundamental properties must the system exhibit if it is to be successful?". This of course will drive the analyst to procure a set of very high-level strategic goals the organization pretends for its system, reflecting therefore the application domain.

If one would look at the "Via Verde" case study, there would be several concerns derived from some of the stakeholders like the customers, the banking entities or even the manufacturers of the parking lot machines. These concerns are listed in Table 1-1 and Table 1-2 according to the several stakeholders and having in mind the fact that their total number should not amount to more than 7.

Table 1-1 Stakeholders of the "Via Verde" System

Stakeholders
Customer
Bank
ATM
Machine Manufacturer

Table 1-2 Concerns of the "Via Verde" System

Concerns
Safety
Multi-Access
Response Time
Availability
Security
Compatibility

Having identified the concerns that relate to a specific problem it is then required that one elaborates them in order to enable their direct influence on the requirement elicitation activities that ensue. This elaboration is achieved in two stages: concern questions and external requirements.

The first serve as safeguards against non-compliant requirements and can be seen as a preliminary test to every requirement that may be elicited in the future.

However, these do not cover sufficient information as to guarantee the detection of all incompatible requirements. Therefore, each concern should be decomposed into external requirements that relate to the different sub-problems that each concern will affect, i.e. seeing as PREview concerns are transversal to viewpoints, their span of influence envelops several areas of a system; however, each sub-system is affected by a concern in different fashions and in several specific manners.

For example, Table 1-3 shows the tabular representation of the decomposition of the Response Time concern. It is clear from the several requirements that there is some distinction between them, namely that some relate to issues of validation, others to issues of information display and still others relating to physical aspects. When it comes to Response Time in the "Via Verde" system it is then safe to conclude that each of the requirements focuses on a specific hazard of the system, and thus the future requirements that derive from viewpoint specification must comply with these in their several areas of interest.

Table 1-3 Response Time Concern

Concern Name	*Response Time*
Requirements	1. The system needs to react in-time in order to:
	1.1. read and validate the gizmo identifier;
	1.2. turn on the light (to green or yellow) before the vehicle leaves the gizmo indentifying area;
	1.3. display the amount to be paid before the client leaves the parking lot;
	1.4. manage entrances and exits from the parking lot.

After having defined the driving concerns of a system, the analyst should then begin to identify the system's viewpoints, what could prove to be a difficult yet crucial task. The process of identifying viewpoints begins with defining their sources and foci, studying their scopes for redundancy and re-examining them for possible changes. Obviously it is an iterative process that extends until a stable set of clearly defined viewpoints is obtained.

In the "Via Verde" case study there was sufficient knowledge of the domain and therefore the hierarchy did not provide additional input, thus the identified viewpoints were:

- ATM;
- Vehicle;
- Banking Entity;
- Entry/Exit Machine;
- User;
- Parking Lot Employee;
- Gizmo;
- Accounting Department.

Although this list seems large, several of these viewpoints will correspond to just one or two tasks, i.e. requirements, and thus this set is not considered unmanageable. An example for

viewpoint description can be seen in Table 1-4 that describes the Entry/Exit Machine viewpoint, following the structure established when defining the PREview viewpoint but withholding for now the information regarding its requirements. Furthermore, if one were to analyze the several foci of all the system's viewpoints it should be apparent that the full scope of the system is contemplated (although foci analysis per se may not guarantee completeness).

Table 1-4 Description of the Entry/Exit Machine Viewpoint

Name	Entry/Exit Machine
Focus	Machine Responsible for controlling parking lot access and gizmo validation.
Sources	Machine Manufacturer, Customer
Concerns	Response Time
	Availability
	Compatibility
	Safety
	Multi-Access

In possession of a clearly defined set of viewpoints the analyst can finally begin to elicit each viewpoint's requirements by interviewing or studying the viewpoint's sources or even by outlining several system models to better understand each point of view concerning the application's tasks. In this stage it is important that each viewpoint has more than one source so as to obtain the greatest level of requirement refinement possible. The iterative nature of the research process will automatically sort out inconsistencies by approaching each source with a previously elicited set of requirements, i.e. a "no blank-sheet" policy.

Additionally there might be a need to decompose a viewpoint into several sub-viewpoints to capture specific problems, either because the elicited requirements lack cohesiveness or if there are conflicts at this early stage. It should be referred that solving conflicts at this stage is much more cost-effective than leaving them for later stages.

Again taking a look at the "Via Verde" case study and maintaining the approach concentrated on the Entry/Exit Machine Viewpoint (since it is the most interesting one due to its span of different functionalities), the several elicited requirements should be detailed according to the

structure established when defining the PREview viewpoint and using the tabular description method adopted by this approach.

Note in Table 1-5 that each of the viewpoint's requirements has a unique identifier as well as a description of its nature, this way any future trade-offs will be achieved in an informed and unequivocal manner. Furthermore, the table's structured nature assists in segmenting each requirement to its specific area of interest without affecting the understandability of the representation.

Table 1-5 Entry/Exit Machine Viewpoint

Name	Entry/Exit Machine
Requirements	1. The Machine detects a vehicle's gizmo.
	2. The Machine attempts to validate the gizmo.
	2.1. If the gizmo is valid:
	2.1.1. The Machine turns on a green light.
	2.1.2. The Machine opens the toll gate.
	2.1.3. The Machine closes the toll gate upon detection of the vehicle's passage.
	2.2. If the gizmo is not valid:
	2.2.1. The Machine turns on an orange light.
	3. Upon exit:
	3.1. The Exit Machine calculates the amount to be paid according to the price range.
	3.2. The Exit Machine displays the amount to be paid.

2.3.5.2 Requirements Analysis

In this phase the idea is to identify non-compliant requirements and correct them, i.e. an analyst should detect which viewpoint requirements conflict with the concern questions, external requirements or even other viewpoints' requirements and guarantee that the whole set conforms to these standards.

In truth, the concern questions should perhaps be applied just before this stage in the PREview process, that is, they should serve, as was previously stated, as a preliminary barrier. Therefore, this stage relates more closely to the consistency checking between viewpoints' requirements and the external requirements of each concern.

First of all, each viewpoint must be checked for internal consistency regarding its requirements; only after their consistency is guaranteed should an external analysis be performed. In [25] and [24] this consistency check is achieved by means of Interaction Matrices, listing an artifact's requirements in one axis and the other artifact's requirements in the other axis, thus allowing for a detailed observation of each requirement; the term *artifact* in this case relates to both inter-viewpoint comparison and viewpoint versus related concerns.

Table 1-6 represents an interaction matrix for comparing two given artifacts and their requirements. Analyzing the several cells one may encounter three types of values: 0, 1 and 1000. These values indicate the degree of the requirements' intersection as, namely and respectively, *independent*, *conflicting* or *overlapping*. Independent requirements obviously do not require further analysis, however, overlapping requirements should be analyzed to decide whether they should be simplified and conflicting requirements should be discussed to decide which of them should be resolved.

Table 1-6 Interaction Matrix for Generic Artifacts

		Artifact2			
		Req2a	Req2b	Req2c	Req2d
Artifact	Req1a	0	0	0	0
	Req1b	1	1000	1	0
	Req1c	0	1000	0	1

Instantiating Artifact2 to a Concern, only requirements req1b and req1c should follow into the negotiation stage, since Concerns are considered as irrefutable. However, in inter-viewpoint comparisons, i.e. considering Artifact2 as a Viewpoint, besides the above mentioned requirements, all of Artifact2's requirements should join them.

2.3.5.3 Requirements Negotiation

PREview does not provide a defined method for the management of inconsistencies and redundancies identified by the requirements analysis phase. Requirements negotiation is left to the judgment of the analyst and the various sources; they begin with the results of the previous stage of analysis and try to resolve the conflicts and inconsistencies, ending in a recommendation that is fed back into the requirements elicitation stage in order to ensure their compliance with concern related constraints.

2.3.6 Summary

PREview's defining trait is its ability to consider both a refined notion of viewpoint and also a definition of an organizational concern. This fact allows for a better guided requirement elicitation process, with concerns as drivers in requirement discovery, as well as generating useful byproducts for later stages of the application's development.

Regarding the PREview notion of a viewpoint, it is clear that PREview viewpoints are flexible, generic entities which can be used in different ways and in different application domains. Furthermore, PREview contemplates both types of viewpoints: those associated with system stakeholders and those associated with organizational and domain knowledge.

PREview basically constitutes a lightweight approach to requirements engineering that can be introduced into existing design processes and adjusted to each organization's standards and modus operandi.

All of these characteristics justify choosing PREview as the representative of viewpoint oriented approaches in an attempt at a hybrid approach involving goal oriented methods; its lightweight nature and the purposeful intention for its easy integration with other processes deems it a perfect candidate.

Chapter 3
Related Work: Goal-Oriented Requirements Engineering

This chapter outlines several of the Goal-Oriented (GO) approaches currently existent or strongly referenced in current work. An introduction to GO methodology is presented, including definition of the generic concept of a *goal* and advantages in its use. Existing methods are then described and its current uses outlined, with a particular emphasis on the KAOS approach.

3.1 Main Concepts

When one is analyzing a project statement, it is the nature of the human mind to automatically focus on the objectives of the statement, i.e. several words almost immediately become imprinted on one's thoughts as an answer to the question "*what is this supposed to do?*". These objectives are called the *goals* of the project and have long been seen as important components in the requirements engineering process.

Goal Oriented Requirements Engineering therefore studies the use of goals for requirements elicitation, specification, analysis, negotiation and many other purposes. The process begins

by analyzing the system up for consideration regarding its several organizational, operational and technical aspects and listing the detected problems and opportunities as goals to be achieved through the ensuing requirements [13].

This is a practice that comes naturally to analysts and that relies on the uniform definition of goals, requirements and their relationships in order to construct a model of the intended system.

3.1.1 What is a Goal?

A *goal* is an objective the system under consideration should achieve [13]. They represent properties that the system is intended to ensure and may be present at several levels of abstraction, from high-level, strategic concerns to low-level, technical concerns [13]. Goals also vary in type, ranging from representations of functional properties the system must offer to non-functional concerns related to quality of service [5].

The system under construction will be constituted by passive components (the software and its environment) as well as active components such as humans, devices and software [13]; these active components are known as *agents*. Goals, unlike requirements, may depend on several collaborating agents in order to achieve satisfaction, being that a software goal that relies solely on one agent becomes a system *requirement* and an environment goal that also relies on just one agent becomes an *assumption* and may not be enforced.

3.1.2 Why use Goals?

Goals represent an important step in handling requirements in the RE process by:

- providing a rationale for requirements, linking high-level strategic objectives to low-level technical details;

- guaranteeing a precise criterion for assessing requirement completeness;
- ensuring requirement relevance evaluation regarding their use in the proof of a goal;
- providing a natural refinement mechanism for easier requirement identification and future readability;
- considering alternative goal refinements that consequently allow exploring different system proposals;
- dealing with conflicts and solving them naturally through further refinements;
- separating volatile from stable information as separate requirements, without dissociating them from the goals they relate to;
- driving the requirements identification process relating to the requirements that support them.

3.1.3 Summary

Goals are useful tools to capture, at different levels of abstraction [13], the several properties the system in design should reflect. Their recognition is almost, at least at their higher levels of abstraction, immediate, constituting perhaps an approach to requirements engineering with a less pronounced learning curve when comparing it to the viewpoint-oriented "point-of-view".

Furthermore, goals provide rationale for requirements, assess their completeness and relevance and are drivers of the requirement identification process.

3.2 Existing Methodologies

3.2.1 NFR Framework

A Non-Functional Requirement (NFR) is a definition of how the system should satisfy its Functional Requirements, i.e. a constraint/directive for the system properties. It usually is subjective in nature and impact and is highly influential on the quality of the final product and its acceptance by the customer.

NFRs strongly condition system architecture and implementation choices besides assisting in uncovering further system requirements that otherwise would remain unseen until latter stages in the software's development. They reflect qualities the system should possess, such as *security*, *accessibility*, etc.

To deal with these issues [6] introduces the NFR Framework, consisting of 5 main components [5]:

- Softgoals
- Interdependencies
- Evaluation procedure
- Methods
- Correlations

Softgoals represent a system's non-functional requirements according to 3 types: *NFR softgoals, operationalising softgoals* and *claim softgoals*. The first are a result of the analysts' study of the domain and the system to be constructed, representing the high-level non-functional requirements of the application.

Operationalising softgoals constitute more specific and concrete solutions (design or implementation related) and are either reduced scoped refinements of the higher-level softgoals, seeing as the firsts' satisfaction depends on their accomplishment, or alternatives for the realization of their parent softgoals.

Claim softgoals translate domain knowledge, analysts' previous experiences and customer demands in order to justify certain softgoals refinements or interdependencies, basically providing a rationale for decisions regarding design and development.

This refinement of softgoals is achieved by means of *IsA relationships*, relating a softgoal to its children, which in turn *contribute* to their parents in a positive or negative way, that is, their completion either assists or hurts their parents' satisfaction. The refinement process is guided by a set of **methods** specific for each softgoal type: *decomposition methods*, *operationalisation methods* and *augmentation methods*. The ensuing softgoal hierarchy should be catalogued and serve as a roadmap for specific softgoal refinement, that is, according to a pre-defined catalogue and the decomposition methods, a security softgoal for an aspect of the application would be refined according to the generic softgoals hierarchy into, for example, availability and confidentiality softgoals [5].

Softgoals, their types and their interdependencies constitute what is called a Softgoal Interdependency Graph (SIG) as displayed in Figure 3-1, which will be **evaluated** to determine whether the root NFR softgoal is satisfied regarding its specification. In these graphs, for each softgoal (represented by clouds) or respective descendants, exist suitable operationalisations (represented by bolder bordered clouds) that provide them with several solutions for their satisfaction. The evaluation procedure begins at the lowest-level operationalisations and follows a bottom-up path according to the types of contributions and refinements (AND - an arc between the connectors - and OR - two arcs between the connectors), resulting in the *satisfaction* or *denial* of the top higher-level softgoals.

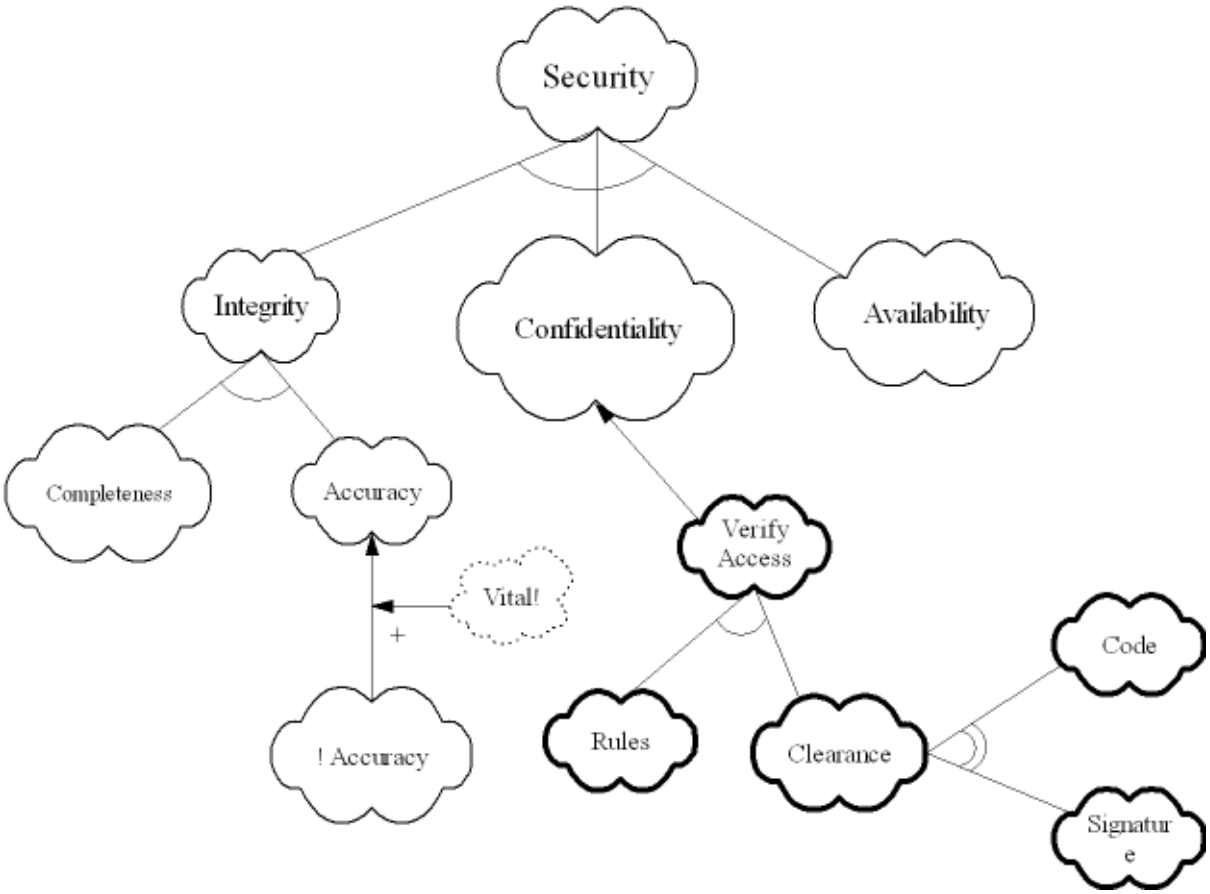

Figure 3-1 Softgoal Interdependency Graph (SIG)

The final major components of the NFR Framework are *correlations*. As it is obvious, non-functional requirements do not exist isolated from the remaining qualities of the system, therefore the several softgoals (generic or operationalising) may relate to one another, reflecting the cross-impact of a system's concerns; these relationships are called *correlations*. Their introduction into the several SIG is parallel to the decomposition and operationalisation processes and represents considered ambiguities, trade-offs or priorities among the SIG elements.

After completion of the initial SIG, its impact should be evaluated and the process repeated until achievement of a satisfactory result. The final SIG should be related to its appropriate functional requirement, representing a *design decision* and the lowest level *operationalisations* related to the functional requirement's specifications, thus providing clear links between functional and non-functional requirements.

This framework clarifies subjective needs of an application into functional requirements, revealing hidden lackings and reutilizing available "know-how". Also, by cataloguing the entire process during the process itself, used refinements end up contributing to the already available "know-how" without the added weight of an *ad hoc* analysis. It is then safe to assume, that this approach contributes *a priori* to the increase of the final product's quality.

3.2.2 GBRAM

Goal-Based Requirements Analysis Method (GBRAM) was proposed by Annie Antón in 1996 [2] and focuses on the need for a clear way of identifying and elaborating goals for requirements specification, since at the time most approaches were based on the premise that the elicited goals were supplied to the analyst.

The approach relies on the following concept definitions for its components:

- A *goal* is a high-level objective of the business, organization or system [2] that focuses on the motivation behind certain system properties and provides a guide for the software analysis process;
- A *requirement* is a specification of how a goal is to be achieved by the system under elaboration;
- An *operationalisation* is a process that elaborates a goal in order for its sub-goals to have clearly defined operational properties;
- An *agent* is the entity or process that seeks to realize a goal, assuming the responsibility for its achievement or lack of it;
- A *constraint* is a requirement that must be satisfied by a goal and represents conditions imposed on its achievement;
- *Goal decomposition* is the process of goal refinement that enables for easier comprehension of the goal itself by defining clear steps/tasks for its achievement and representing them as sub-goals;

- *Scenarios* are descriptions of certain system properties that may arise from constraints or restrictions;

- *Goal obstacles* are impeditive behaviors represented by certain goals that pose a threat to other goals' completion.

Furthermore, a goal belongs to two different types:

- *Achievement goals* represent organization objectives that translate into functional properties of the system under analysis;

- *Maintenance goals* represent conditions to be satisfied by the system through constraints imposed on the system itself and tend to translate into non-functional requirements.

The GBRAM process is divided in two stages: *goal analysis* and *goal evolution*.

Goal analysis focuses on the extraction of information from the available sources (project statements, diagrams, etc) in order to identify the several system goals, or at least the higher-level ones. Several sources are advised for unequivocal goal identification and clear goal elaboration, being also suggested the focus on key action words when analyzing stakeholder descriptions.

This stage also serves to identify the several intervening agents, additional stakeholders and constraints. The several agents are identified by searching for the entities responsible for each goal's realization and the constraints, seeing as they result from conditions imposed on the system, are identified by searching for key temporal connectives. Finally the goals are classified according to the conditions they represent as *achievement* or *maintenance* goals.

Goal evolution consists in the refinement and operationalisation of system goals. Since stakeholders may often change opinions over time, goals are likely to be subjected to constant changes and improvements. Therefore it is essential to begin by elaborating each goal through goal obstacles identification, scenario analysis, constraint definition and goal operationalisation.

Obstacles represent the conflicts that may arise between goals or the presence of impeditive conditions, whereas scenarios facilitate the description of priorities that might ensue from obstacle analysis.

Goal refinement translates into the merging or splitting of goals according to the existence of redundancies or the need for goal clarification, respectively. Regarding goal clarification, this results in the decomposition of goals into clearer sub-goals that specify steps in the parent goal's realization. Refinement also includes the identification of constraints imposed on the several goals and the operationalisation of goals, which in turn result on the imposing of pre and post conditions.

The GBRAM approach therefore defines a top-down analysis method for refining goals and attributing them to agents starting from inputs such as corporate mission statements, policy statements, interview transcripts etc. and providing an elaborate yet understandable analysis of the several system goals and the concepts implied in their definition.

3.2.3 I* Framework

The I* Framework was developed in 1995 by Eric Yu [26] and proposes an *agent based* approach to requirements engineering. By *agent based* it means that the approach focuses on the use of the intentional actor as the central construct, revolving around the different system stakeholders and their relationships. Agents relate to other agents in order to achieve *goals*, carry out *tasks* and obtain *resources*, establishing a connection between *depender*, *dependum* and *dependee*.

A *depender* is an agent that relies on another agent for a goal, task, sub-goal or resource, which in turn is the *dependum*, and the agent dependent upon is the *dependee* [5], as shown in Figure 3-2. The representation of a system according to these types of relationships allows for a better understanding of the social networks implied in the system itself and the necessary interactions required for the realization of certain system properties.

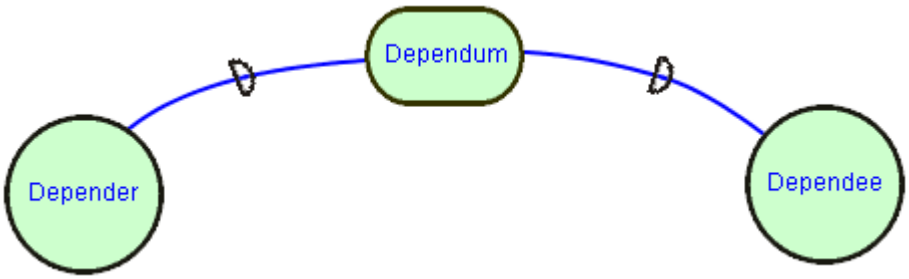

Figure 3-2 Goal Dependency

The driving force behind this social analysis is the question "why?" regarding the purposes of the several requirements and regarding their types. It is many times useful to understand the motivations and rationales behind the several activities, thus gaining a better perception of the system for the design of the software processes.

I* proposes two stages in its approach, each with a graphical representation that makes it understandable for dialogues with the several stakeholders: the Strategic Dependencies (SD) Model and the Strategic Rationale (SR) Model.

In the SD stage, the purpose is to study agents and their dependencies, focusing on what is gained or lost by the achievement or failure of a particular dependency. To be considered is the particular nature of each agent, that is, each agent is an *intentional* actor with a drive to fulfill certain commitments, with an inherent *autonomy* and free will, existing within a network of *social relationships* with other agents, with a distinct *identity* and defined *boundaries* and with the ability to *reflect* upon its actions and strive for *self-achievement*.

The process begins by identifying the system's agents according to the characteristics listed above as well as the higher-level goals. The corresponding model includes the previously referred types of *dependum* as well as the dependency links between the several agents [5]. Regarding the types of *dependum*:

- A *goal* is a system objective, a property that the system should reflect, and can be achieved in multiple ways;
- A *softgoal* is akin to the goal but with a degree of *fuzziness*, i.e. it can represent a non-functional property or a less clear goal;

- A *task* is the equivalent of a requirement and corresponds to a *dependum* whose manner of achievement is clearly defined;
- A *resource* is a *dependum* that is forwarded between agents, a result to be shared.

In the SR stage, the previous analysis is elaborated regarding each agent by deepening the degree of the driving questions why and how, as well as considering alternative solutions for the higher level problems. This goal decomposition mechanism is similar to that of the NFR Framework [6] in a way that each agent evaluates the goals that he is associated with and devises methods for accomplishing them. The resulting model possesses new types of links reflecting relationships akin to the NFR *correlations* and *contributions* and basically extends the previous model with the rationales for the already present dependencies.

Across these two stages, the I* Framework provides several alternative dependency structures and paths for goal completion, as well as different agent cooperation strategies, that after careful analysis may be chosen for realization [5]. The different models reflect the intentions of the agents and the higher-level goals that compose the system functionalities, thus highlighting the importance of agents, their specifics and dependencies in the process of software design.

3.2.4 GRL

GRL (Goal-oriented Requirement Language) is a language used in agent-oriented and goal-oriented modeling and reasoning of requirements, focused in dealing with non-functional requirements [15]. It is strongly based in the i* and NFR frameworks for specifying non-functional requirements and is part of the URN (User Requirements Notation) along with Use Case Maps.

GRL, like the frameworks it is based on, relies on components to represent the several concepts involved in requirements engineering, organizing them in three categories: *intentional elements, links*, and *actors*. The intentional elements comprise *goals*, *tasks*, *softgoals* and *resources* and their intentional nature relates to the fact that their use aims at explaining particular system behaviors, structural aspects and design alternatives, along with the used criteria.

As with the i* framework, the analyst that uses GRL is primarily concerned with explaining those structural options or any constraints that might be introduced, regardless of their concrete nature (softgoals are fuzzy by nature). Implementation details are scaled back in order to allow the analyst to assume a strategic stance and observe the higher-level architectural alternatives aiming at a current yet extensible model that considers the surrounding environment.

However, GRL defers from i* by not allowing agent specializations and offering constructors for enabling relationships with external elements: *non-intentional elements* and *connection attributes*. GRL also offers additional elements that aim at argumentation and contextualization such as: *beliefs, correlations, contribution types* and *evaluation labels*. Some of these elements are recognizable from the NFR framework and contribute with the specification of satisfaction states, extending the types and ranges of qualification of the i* relationships.

The GRL model can be composed of a global goal model or a series of distributed goal models for several actors, being that if more than one actor is present, inter-actor dependencies should also be present [15].

GRL components are defined much like the approaches they derive from:

- A *goal* constitutes a condition that the stakeholders want the system to reflect and can be specified as a *business goal*, reflecting a state of affairs the stakeholders wish to achieve, or a *system goal*, i.e. a functional requirement of the application;
- A *task* details the method for satisfying its parent node, specifying a particular way to achieve either the goal, the sub-goal or the higher-level task it relates to;
- A *softgoal* is similar to a goal but lacks the precise criteria for assessing its achievement. It is subjective in nature and represents a NFR in the system under consideration;
- A *resource* is a physical entity that is forwarded between actors considering its availability.

In the GRL model, the several components are connected by links of several types:

- *Means-ends* links are used to connect tasks to goals, offering the tasks as alternative methods for the goal completion;
- *Decompositions* define what other sub-elements need to be achieved or available in order for a task to be performed;
- *Contributions* represent effects from one component to another;
- *Correlations* allow for expressing knowledge about interactions between intentional elements, encoding such knowledge;
- *Dependencies* relate actors and components as *depender*, *dependum* and *dependee*.

A global GRL model is basically a set of goal model structures connected by correlation links, allowing for a better understanding of the transversal nature of NFRs, while observing the system from an agent oriented perspective; all of this with the benefit of the same graphical notation as that used in the i* framework.

3.3 KAOS

The KAOS (Knowledge Acquisition in AutOmated Specification) approach stems from cooperation between the Universities of Oregon and Louvain that began in 1990, under the supervision of Professor Axel van Lamsweerde. It was initially proposed in forums of Artificial Intelligence but it was quickly found to have potential regarding applications of the machine learning domain to requirements engineering.

In [13] Lamsweerde et al. state that requirement analysis is "made of two coordinated tasks": *requirements acquisition* and *formal specification.*

The first relates to the structuring of requirements into a preliminary model of the system, elaborated and expressed in a "rich" modeling language. This language encompasses a set of concepts required to provide an adequate description of the system to be designed such as objectives, constraints, entities, relationships, etc. and should balance enough formality as to enable a formal basis for the elicitation of requirements with the ease of understanding as to be readable by clients.

The task of formal specification relates to the refinement of the requirements model into more precise one, relying on a level of formalism necessary for detailed formal verifications and the generation of prototypes. However this second task is not the focus of this work and is therefore only briefly referred to.

Compared to other approaches, KAOS is the only one that allows formal specifications and provides a full fledges commercial tool [21]. In fact, this tool called Objectiver has been used in several industrial projects and is currently being used by some of the main players within the European aerospace industry relating to a project aimed at improving air transportation security. The European project is known as SAFEE [1].

3.3.1 The KAOS Method

KAOS as an overall approach is divided into three components: a *conceptual model* for requirements acquisition and structuring with the respective acquisition language; a *set of strategies* for the elicitation of requirements and an *automated assistant* to provide the guidance required for the acquisition process corresponding to the chosen strategy.

The conceptual model is a meta-model intended to provide a basis for the requirement acquisition language and provide abstractions for the requirement models. It allows for functional and non-functional requirements representations and is divided into three levels: the *meta-level* that provides the sufficient abstraction for representing the requirement models, the *domain-level* that maps the higher level abstractions into the specific concepts pertaining to the application domain, and the *instance-level* that refers to specific instances of domain-level concepts [13]. This meta-model is essential since it drives the knowledge acquisition process [5] and defines the concepts on which the acquisition strategies will base themselves upon.

The acquisition strategies define methods of traversing the conceptual model as steps for instancing the several meta-model components. For the purposes of this work a goal-directed acquisition strategy is considered and results in a specific combination of detailed steps that will result in a well-defined requirements model. This strategy will be detailed in further sections.

Regarding the automated acquisition assistant, it comprises of a support strategy for the acquisition process itself, based on two repositories: the *requirements database* and the *requirements knowledge base*. These repositories are updated along the development process, the first with up to date requirement models for the strategies under consideration, and the second with domain knowledge organized into hierarchies for easy maneuverability.

This work will present the main concepts involving the goal-directed requirements acquisition strategy, as well as the specific models it ensues, from a meta-model level (knowledge structures) to an instanced level represented by the goal decomposition trees and its "brethren" models.

3.3.2 The KAOS Process

The instancing of the KAOS acquisition process into a goal-directed strategy is defined in [13] as having 7 steps that may overlap, iterate and require backtracking:

1. Identification of Goals, their structures and concerned Objects;
2. Identification of Agents and their Capabilities;
3. Operationalisation of Goals;
4. Object and Action Refinement;
5. Derivation of strengthened Actions and Objects to Ensure Constraints (operationalisations);
6. Identification of alternative Responsibilities;
7. Assignment of Actions to responsible Agents.

These steps, as is said previously, may be iterated, while overlapping, and may require backtracking regarding later changes. For the purposes of this work, these steps will be instantiated into KAOS models as defined in [21] based on the case study enunciated in a previous section, and furthermore, step 5 will not be considered since it relates to the formal specification of the several requirements which, as previously stated, is not the focus of this work.

3.3.3 KAOS Goal Model

The KAOS Goal Model is the set of interrelated goal diagrams necessary to tackle a particular problem and is comprised of a number of entities and relationships whose use in its construction enables to represent *how* and *why* a goal is achieved:

- **Goals** are the focus of this model and represent objectives to be met through agent cooperation, prescribing a set of behaviors the system is supposed to reflect.
- **Sub-Goals** are goals that are linked to other goals through means of refinement relationships, contributing to the satisfaction of the goal they refine. Meeting the conditions of all sub-goals should automatically entail the satisfaction of their "parent" goal.
- **Agents** are humans or automated components that are responsible for achieving certain requirements and/or expectations.
- **Requirements** are low-level types of goals whose achievement constitutes a responsibility of a given software agent.
- **Expectations** are goals assigned to agents that interact with the system; they reflect interactions between the system and its environment and their achievement is not a system responsibility.
- **Domain Properties** are assertions about certain objects of the software environment enunciated as domain invariants or hypothesis, i.e. properties that are known to hold in all states of a domain object or properties that are supposed to hold, respectively.
- **Obstacles** are certain conditions that prevent the achievement of system goals. The definition of these undesired behaviors represents a defensive approach to software modeling.
- **Refinement Links** are relationships between a goal and its sub-goals that represent the decomposition of an objective into clearer steps in its achievement.

- **Responsibility Links** represent the connection between a software agent and the requirement whose achievement it is responsible for.
- **Assignment Links** represent the connection between an environment agent (one that interacts with the system) and the expectation whose achievement it is responsible for.
- **Obstruction Links** relate obstacles to goals, representing the impediment an obstacle represents to a goal's satisfaction.
- **Resolution Links** relate goals to obstacles, representing solutions to the presented impediments.

All of these elements can be seen in Figure 3-3 and represent the several decompositions, interactions and conflicts that may occur when attempting to achieve the Parking Lot Entrance goal, albeit in a simplified manner.

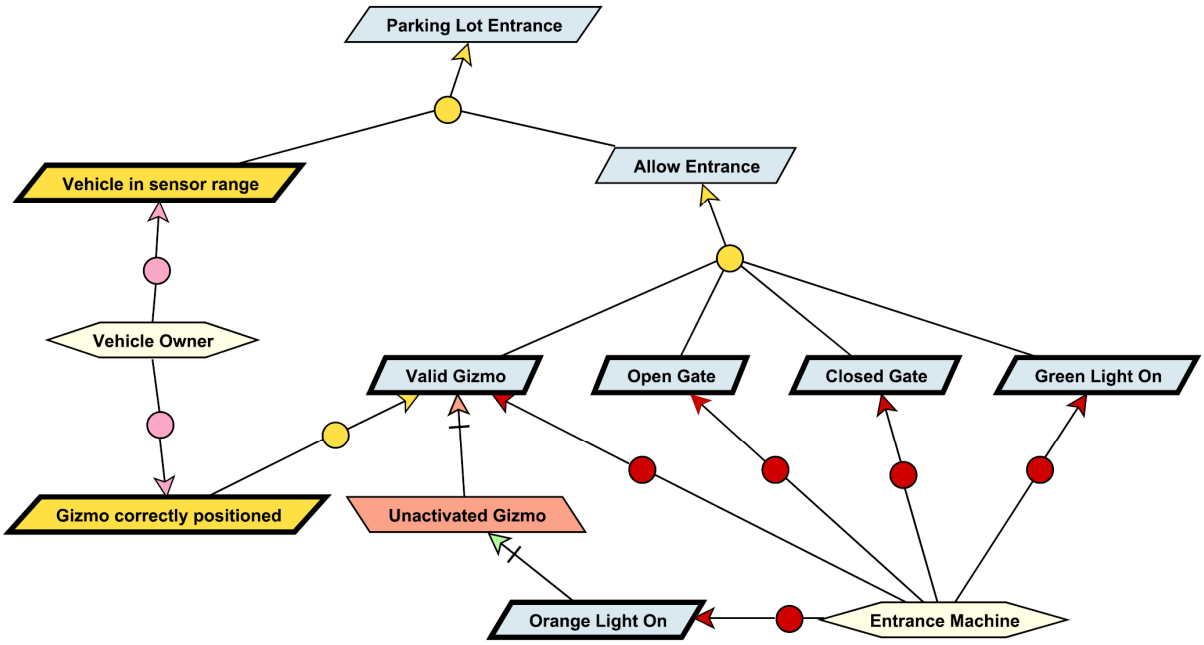

Figure 3-3 Parking Lot Entrance Goal Model

Also worthy of reference is the fact that KAOS encourages the use of requirement patterns when eliciting the several system requirements, this way attempting to solve the "blank page" issue when interviewing stakeholders. Many times, goal models end up being instances of previously defined requirement patterns.

Furthermore, in [21] several completeness criteria are provided for the analysis of the overall system model. The first two pertain specifically to the goal model:

1. *"A goal model is said to be complete with respect to the refinement relationship 'if and only if' every leaf goal is either an expectation, a domain property or a requirement.*"

2. "*A goal model is complete with respect to the responsibility relationship 'if and only if' every requirement is placed under the responsibility of one and only agent (either explicitly or implicitly if the requirement refines another on which has been placed under the responsibility of some agent.*"

3.3.3.1 Conflicting Goals

Conflicting goals is a well-known phenomenon in the world of goal-oriented requirements engineering, and KAOS is no exception. Recognizing this possibility of conflicting goals, the KAOS/Objectiver methodology proceeds to represent them as seen in Figure 3-4. The conflict is identified through analysis of each goal's description, along with essential knowledge of the domain, and it is evidenced in the KAOS Goal model.

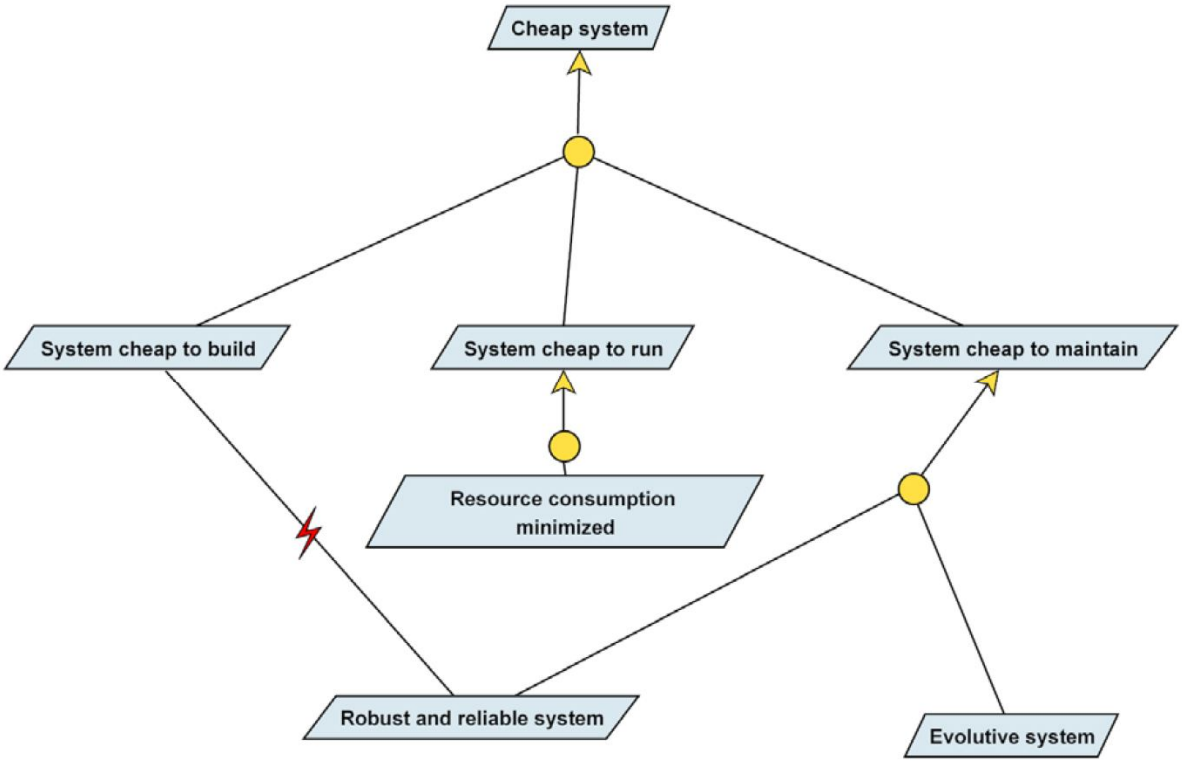

Figure 3-4 Generic Goal Pattern with Conflict

The above mentioned figure details a generic goal pattern with a conflict between two of the goals; by using a generic representation it is clear to understand why those goals would collide - robustness almost always implies growing expensiveness in building the system.

3.3.4 KAOS Responsibility Model

The KAOS Responsibility Model is the set of responsibility diagrams that can be derived from the goal model and displays the requirements or expectations an agent is responsible for. It therefore comprises three of the previously described elements: an **Agent**, its assigned **Expectations** and/or **Requirements**.

Such a model is shown in Figure 3-5 representing the responsibilities of the Entrance Machine agent.

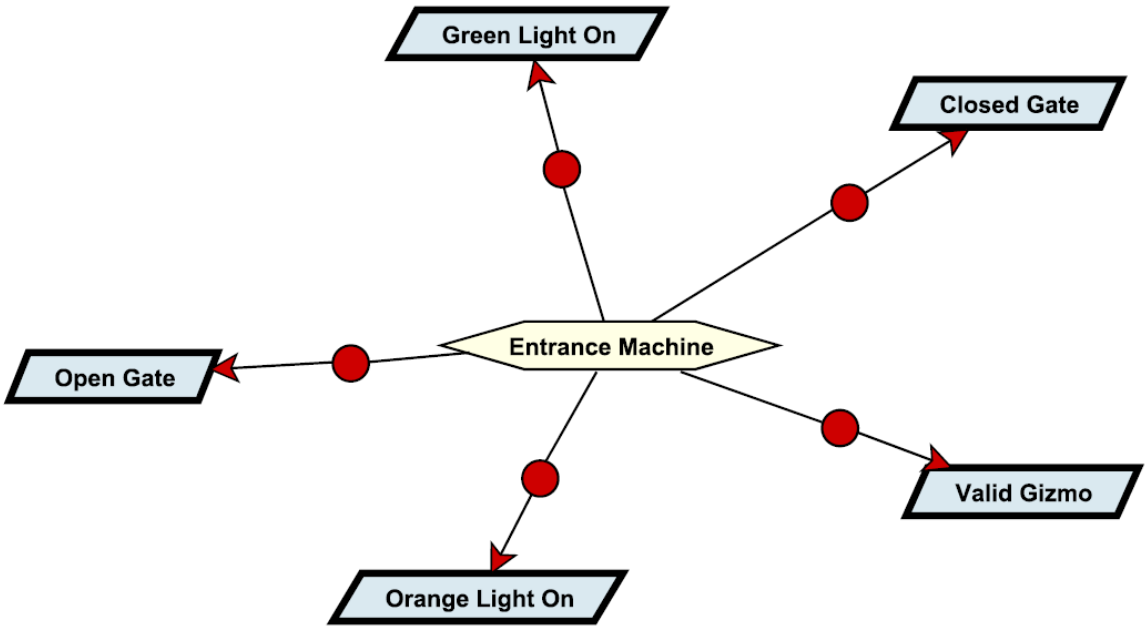

Figure 3-5 Entrance Machine Responsibility Model

3.3.5 KAOS Object Model

The KAOS Object Model defines the several concepts of the domain that may be relevant with respect to the known requirements or that represent constraints on the system itself in order to satisfy requirements.

The objects may be defined as belonging to three types:

- **Entities** are autonomous, passive objects whose definition is independent from other objects.
- **Agents** are active objects that perform operations in order to achieve several types of goals.
- **Associations** are passive objects whose definition depends on the objects they link.

Their identification is parallel to the process of goal identification and definition, or may result from browsing the goal model or even from discovering system components that are

necessary for a requirement's satisfaction. Their representation is compliant with the UML standards for class diagrams.

These objects are connected to the goal model through **Concerns Links**, relating requirements to the objects that are needed for them to be satisfied.

An object model can be found in Figure 3-6, representing the components of the Parking Lot System.

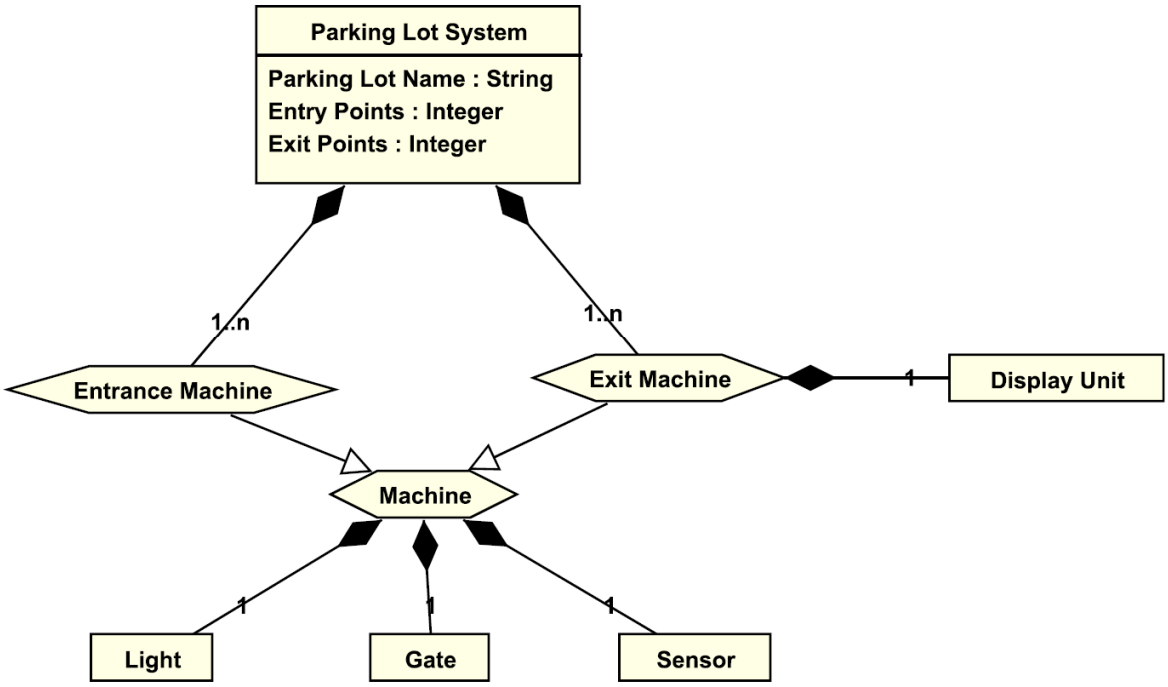

Figure 3-6 Parking Lot System Object Model

3.3.6 KAOS Operation Model

The KAOS Operation Model sums up all the behaviors necessary for an agent to fulfill its requirements. These behaviors are translated into *operations* that work on the previously defined objects, being responsible for their creation, the triggering of object state transitions and the activation of other operations through **events**.

Their elicitation happens during the stakeholder interviews, in case the stakeholders find it necessary to describe certain behaviors in order to define a system goal, or by observing the modeled requirements, representing the operations as how the requirements have to be realized.

The operationalisation (fulfillment) of requirements follows a set of heuristics defined as follows:

- *Static* requirements are translated into *objects*;
- *Dynamic* requirements are operationalized into *operations*;
- \item Requirements that are both *static* and *dynamic*} are operationalized into interacting *objects* and *operations*.

The operation model therefore requires the definition of the following elements (illustrated by Figure 3-7):

- **Operations** are performed by agents and specify objects' state transitions.
- **Events** are ephemeral objects that trigger operations performed by agents and can be external or produced by other operations.
- **Input Links** are established between objects and the operations they serve as input for.
- **Output Links** are established between objects and the operations that produce them as output.
- **Cause Links** relate events to the operations they initiate or terminate.

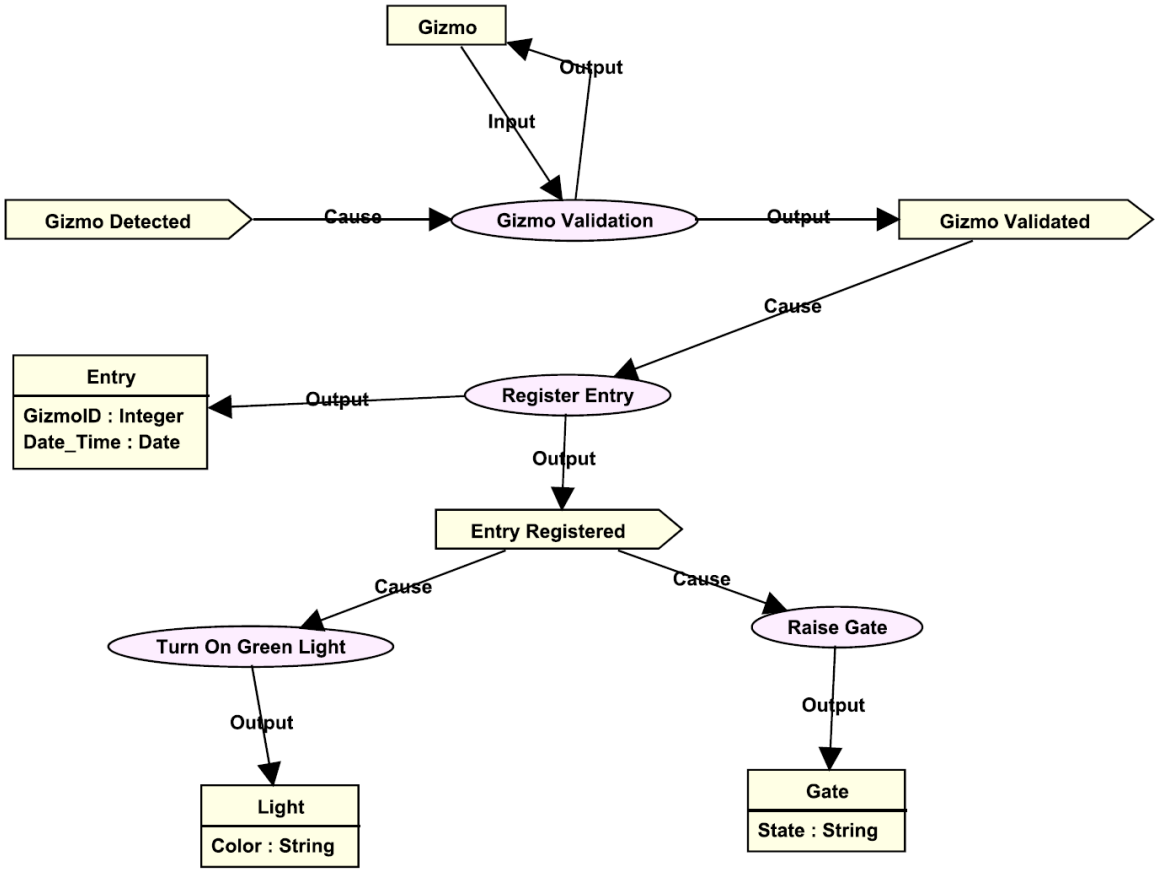

Figure 3-7 Parking Lot Entry Operation Model

For this model, three more completeness criteria exist that define rules for its evaluation:

3. "*To be complete, a process diagram must specify*
    a. *The agents who perform the operations*
    b. *The input and output data for each operation.*"
4. "*To be complete, a process diagram must specify when operations are to be executed.*"
5. "*All operations are to be justified by the existence of some requirements (through the use of operationalisation links).*"

The last criterion reveals a specific trait of the operation model that pertains to the finality of the overall system model; the operationalisation of leaf nodes from the goal model provide a conclusion to each particular branch, guaranteeing its achievement; however, requirement nodes left unoperationalized leave much room for subjectivity.

Furthermore, this model bridges the gap between the problem description and the solution description, adding to the traceability of the several models and to the flexibility of the approach, i.e. an analyst may decide to start from goals or from a set of required operations.

3.3.7 Summary

The KAOS approach, as explained by the Objectiver methodology [21], addresses requirements identification and of the intervening agents by relying on the construction of a requirements model segmented into four types of sub-model: the goal model, the responsibility model, the object model and the operation model. This conjunction of the several models is clearly illustrated by Figure 3-8.

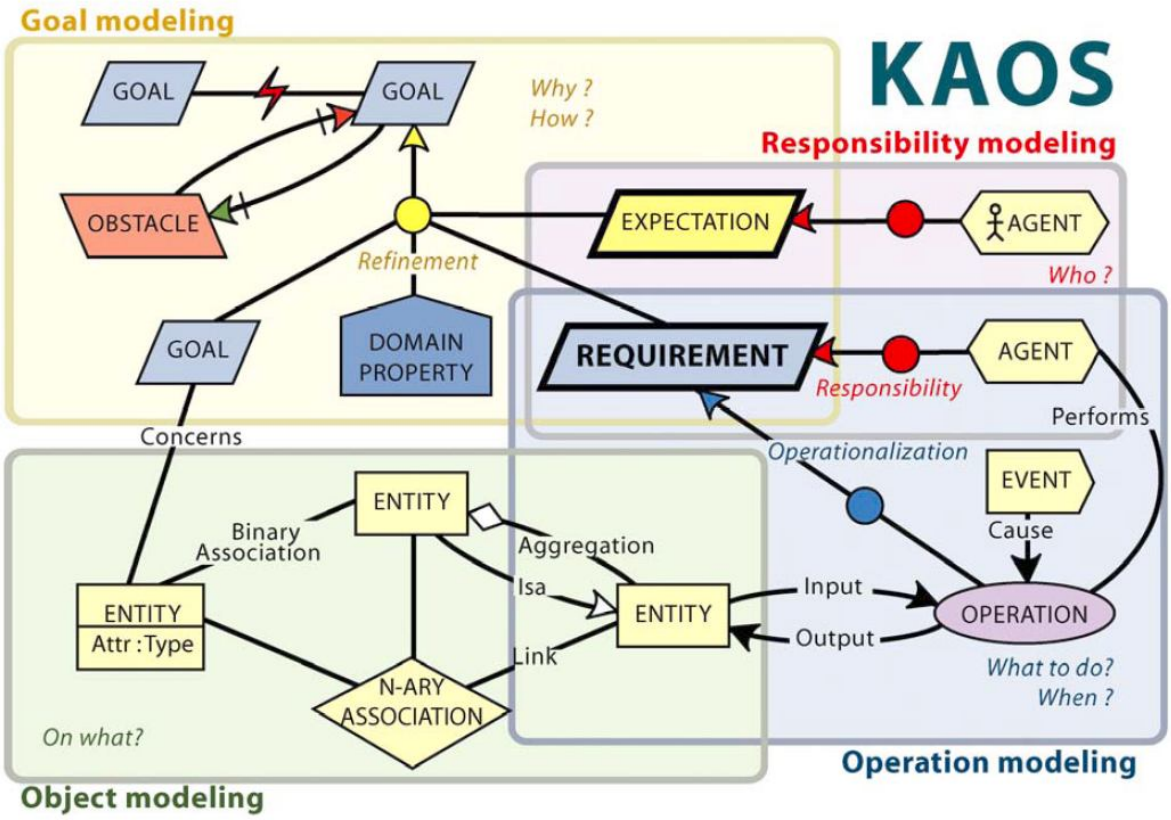

Figure 3-8 KAOS/Objectiver Modeling Methodology

This method of analysis, although presented in this work in a simplified manner, is focused on the problem itself, introducing through means of the several diagrams, important aspects of the application that would not be evident in the problem description.

Furthermore, KAOS addresses the issue of traceability between the problem description and the solution description stages by operationalising the several requirements that require completion. This way, analysts have a means to ensure that their intake on the system is translated into the expected solution, and the developers can have some context surrounding the solution they need to develop.

By defining completeness criteria, KAOS ensures that the completion of the established goals is clearly defined and that every requirement is attributed to an agent, leaving no room for "wishful thinking" [21]. This aspect of the approach is furthered by the formal specification that would ensue, however such is not the purpose of this work.

Due to these particular capabilities and the clearly defined and unambiguous modeling aspects, the KAOS approach using the Objectiver framework provides an almost "elegant" method for approaching requirements modeling and therefore merits choosing it for the basis of the proposed work.

Chapter 4
The Hybrid Approach

This thesis's purpose is to develop the Hybrid Approach using as basis the Viewpoint-Oriented PREview approach and taking advantage of the formal decomposition techniques the Goal-Oriented KAOS approach provides to complement the requirements discovery process of the base approach.

As such, seeing as integrating two complex requirements engineering approaches demands a certain degree of understanding regarding the elements of both approaches and how they relate to each other, if in any way, a comprehensive conceptual model analysis is required.

4.1 Conceptual Model Analysis

A conceptual model is a map of concepts and their relationships that can be used to describe the semantics of an approach and represent assertions about its nature. Specifically, it describes an approach's significant components and provides means to collect information and display characteristics of and associations between pairs of those components.

Concerning the PREview approach, no conceptual or meta-model has been, to this point, put forward, either by the authors, or by others and validated by the original authors; this meant that further work would encompass, at the least, designing a conceptual model of PREview. Regarding the KAOS approach the situation is precisely opposite: much work has been put into designing a KAOS meta-model, since its early days; seeing as it was seen from the start as a formal approach, a meta-model would of course be essential for its definition.

However, the "problem" with the KAOS meta-model is the vast number of concepts it includes, and thus its very complex nature. Therefore, in order to better analyze the approach and its integration with PREview, a certain number of steps were taken when dealing with the aforementioned model:

- as it is described in the following section, this hybrid approach will concern itself merely with KAOS concepts relating to Goal Models, this means that components like operations and events can, at least for now, be put aside;
- even considering just the KAOS Goal Models, the approach's meta-model is still complex enough as to complicate a comparison between both approaches, so some concepts of the Goal Model were "simplified" in order to facilitate the establishing of correspondences.

The simplified meta-models, or conceptual models in order to minimize commitment to a notation, are represented in Figure 4-1, as well as the correspondences found when analyzing both approaches in parallel:

- Viewpoints are encapsulation units, but each of them symbolizes an agent's intake on the system, and the requirements that intake generates; on the other hand, KAOS agents are entities responsible for the fulfillment of requirements and expectations. Therefore, it seems logical to establish a correspondence between PREview Viewpoints and KAOS agents;
- KAOS Requirements and Expectations relate closely to PREview Requirements, whereas PREview's external requirements only relate to KAOS Requirements;
- Both PREview Concerns and KAOS Soft-Goal are non-functional considerations on the system and are orthogonal in nature;

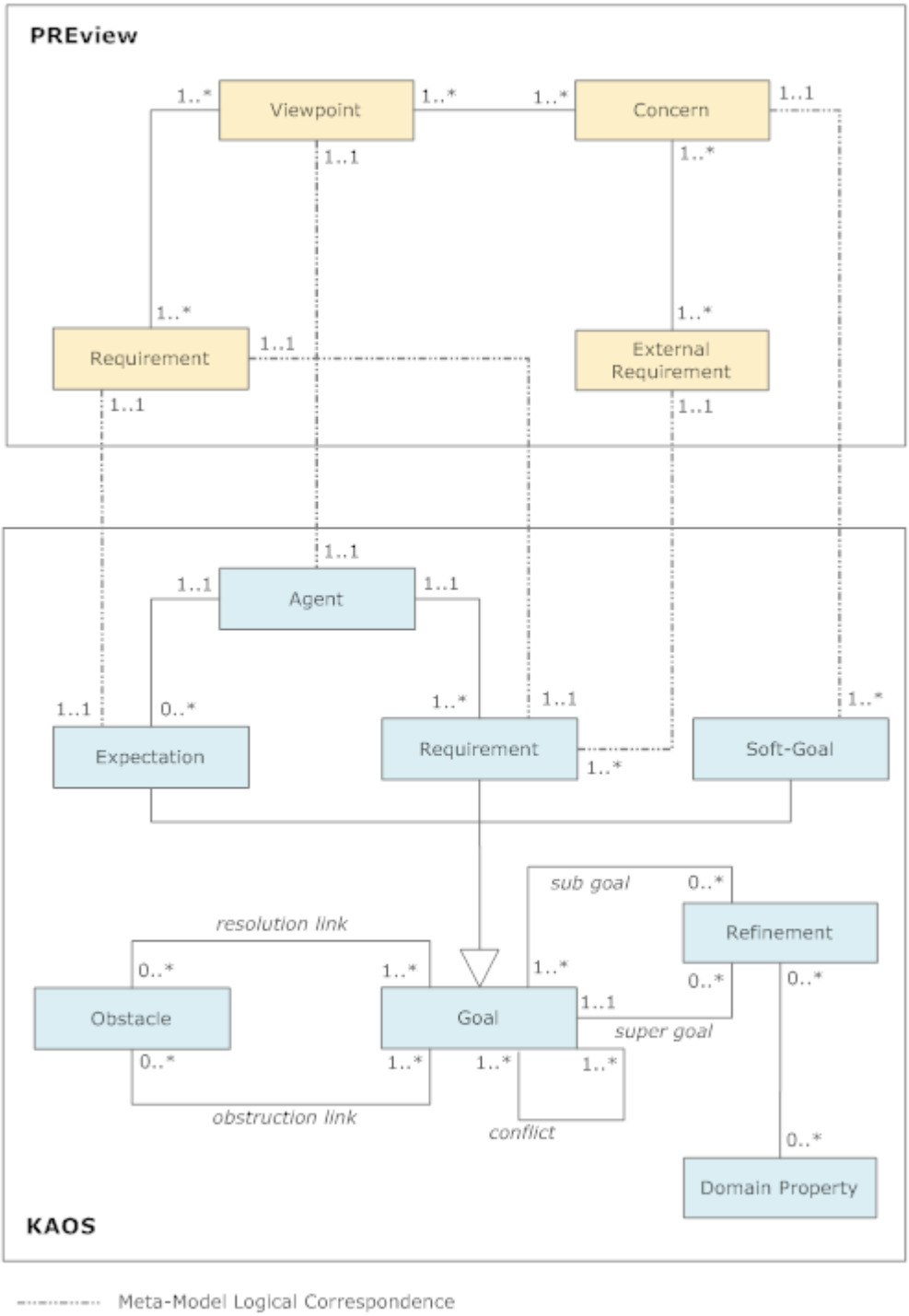

Figure 4-1 Conceptual Model Correspondence

A final concept to keep in mind is the distinction between the nature of a Viewpoint and a KAOS agent: although a relationship can and should be established between them, it is important to distinguish between the position from which a viewpoint observes the system

and that of a KAOS agent, that is, a viewpoint is always foreign to the system, whereas an agent may be a system module.

4.2 Hybrid Approach's Heuristics

Choosing a starting point for this hybrid approach signifies identifying the concessions the PREview approach should make in order to better accommodate the introduction of the KAOS modeling components. This means that although both of these approaches have merit and have been the target of extensive studies and applied to numerous projects, the PREview approach's methodology should be adapted to allow for an easier integration of the KAOS goal decomposition mechanisms, whilst remaining true to the approach's original principles.

Furthermore, the complementary nature of these approaches is due precisely to their specific characteristics as approaches, not due to the elements or steps they entail. And most of all, one should not reinvent the wheel, thus much of what each approach brings to the picture and how it does so, should be taken advantage of, in its "natural state".

That being said, after analyzing the many points in which both the KAOS and the PREview approaches meet and/or complement each other, a set of Heuristics for this Hybrid Approach was created. This set of heuristics is of course largely based on the PREview approach's process and introduces the goal decomposition mechanism of KAOS into the method.

On a summarizing note, there are several reasons for this choice:

1. the first is the fact that PREview in itself was an approach designed to be integrated with other approaches to suit particular needs of certain projects, furthermore, as it was described in the corresponding section, PREview is a lightweight approach that restricts itself to the process of requirements elicitation: this may prove to be useful in these early stages of such an hybridization process in what concerns difficulty management;

2. a second reason is the fact that, in PREview, from Viewpoint identification to requirements identification there is no specific process, no set of rules or guidelines, just some pointers that by themselves are obvious like "consult your sources", "the

more sources, the more information"; this of course is useful advice, but for inexperienced analysts it might leave too much room for misunderstandings and errors;

3. on the other hand, and as a third reason, the KAOS approach provides a systematic method for requirements identification by means of goal decomposition, however in requirements engineering, and keeping true to the spirit of the PREview approach, sometimes a certain degree of flexibility is required.

As it was pointed out in the KAOS approach section, when a developer reads a project statement or gathers the stakeholders for a meeting, the most obvious and normally first artifacts to be produced, be that imprinted on a piece of paper or on the developer's mind, are the system goals.

"The system is supposed to do this." or *"We require the system to produce that."* are common statements, and their translation to system goals is almost immediate.

However this process of identifying a system's goals can sometimes be overwhelming, as is the case when dealing with large and complex systems; such a process would benefit from segmentation into smaller and thus more manageable components, while ensuring that all stakeholders and all environment components are being contemplated. Stepping from the KAOS approach to the PREview approach, this role corresponds to that of the system's viewpoints.

4.2.1 Produce List of System Viewpoints

As it was the case in the PREview approach, this set of viewpoints will correspond to what may be called foreign agents - the system's stakeholders – and of course the system's environment, in what concerns components that may interact with the system itself. As it was explained during the conceptual model analysis, extrapolating this definition to the set of KAOS agents, we will obtain a specific sub-set of these elements relating to strictly non-system components.

Identifying a system's viewpoints in the PREview approach implied a careful and sometimes extensive definition of the items that composed them, especially the viewpoint's focus and its

sources. However, contrary to what was defined in the original approach and in order to find some common ground with the KAOS approach, the process of identifying these “initial viewpoints” should not worry itself with depth, in what relates to Viewpoint description, or scale, in what relates to the number of initial viewpoints.

More precisely, these initial Viewpoints should only possess three attributes: a name, its type (stakeholder or environmental) and a focus (albeit a simple one).

Why such a lax approach at this identification? Mainly for two reasons:

- The first attempt at identifying the system’s viewpoints in the PREview approach, even when done so by an experienced analyst, would probably not output the final set of Viewpoints. Furthermore, the initial set of viewpoints is constantly being revised and modified, merging ones that are too similar in nature or splitting those that do not make sense together, thus if one can obtain a set of identifiable Agents while avoiding the amount of redundant work that goes into describing Viewpoints in depth at every step, one should choose to do so;
- This hybrid approach bases is requirement identification process on the KAOS Goal Models; these models in turn use as subjects what are called Agents. Although without presuming to possess a set of literal KAOS agents, these initial viewpoints would facilitate integration between the PREview Viewpoints and the subjects of KAOS Goal Models.

Regarding the process itself, a few guidelines should be followed in order to direct the analyst in the identification of these initial viewpoints:

- It may help to separate viewpoint identification between types, that is, identifying separately the stakeholder related viewpoints and the environmentally related viewpoints;
- Having in mind that the basis for this work is a project document or the analyst's notes from an interview with the main stakeholders, a method that works well is syntactic analysis:
    - What form do project requirements normally take? "System component A is expected to behave like this" or "Agent B interacts with the system in such a way";

- If one is to study these sentences, one will find that they have a basic syntactic structure of actor + verb or action + object;
- Seeing as a viewpoint relates to the perpetrator or the target of an action, it makes sense that in the several sentences, the various actors or objects may provide precious indications as to Viewpoints that should be regarded.

Considering the Case Study that was used when presenting both the PREview and the KAOS approaches, the "Via Verde" case, the first task would then be to identify the system viewpoints taking into account these proposed guidelines.

As such, if one were to look at the project statement, one sentence that might come up for analysis would be:

"To access a parking lot, a user must place the gizmo in a readable place"

Analyzing this sentence from the stakeholders' point of view, an obvious viewpoint comes to mind: the system's user. From a syntactic standing point, this viewpoint pops up as the actor in the sentence, the entity that perpetrates the action. On the other hand, from a system's environment point of view, the object of the sentence, the gizmo, suggests yet another viewpoint: an agent belonging to the system's environment, seeing as the gizmo plays a part in the system itself.

Extending this analysis to the rest of the project statement, one may achieve the following set of initial viewpoints:

- Stakeholder Related Viewpoints:
  - User;
  - Parking Lot Employee;
  - Banking Entity.
- System Environment Related Viewpoints:
  - Entry Machine;
  - Exit Machine;
  - Gizmo;
  - Accounting Department;
  - Vehicle;
  - ATM.

This division of viewpoints into categories is useful during the identification stage, and will also translate into particularities when verifying the developed goal models.

4.2.2 Develop the System Viewpoints Using Goal Models

Following the identification of the system's viewpoints, the PREview approach suggests that each viewpoint should be described in terms of the requirements it is concerned with when it relates with the system. But as it was previously stated, this step was not immediate, nor was it achieved in a systematic manner by the PREview approach. This is where the KAOS approach may step in.

A system's goals are in fact the main artifacts that come from either reading the project statement or from interviewing the several system stakeholders. By identifying these goals one would be identifying the system's objectives, and by taking advantage of the encapsulating units called viewpoints one would be not only guaranteeing that all interested parties are being contemplated, but also that the several system goals would be attributed to the stakeholders they are concerned with.

Furthermore, when dealing with requirements identification, the KAOS approach takes a path that is precisely the opposite of the one taken by the PREview approach, that is, a clearly defined set of formal rules to identify the system's requirements. KAOS provides a mechanism called Goal Decomposition that takes each goal and divides it into several sub-goals and/or requirements and/or expectations.

This method has several advantages that have already been stated in the KAOS approach section. However, it is always good to underline that by using this method, which by adhering to the KAOS/Objectiver Methodology can be done in a graphical way, completeness is assured. According to it, a goal model is not complete until all leaf nodes are either expectations or requirements; this guarantees that all system goals are decomposed into specific tasks or conditions that may be performed or met by software or environment agents. Furthermore, a goal's completion is assured by the completion of its children.

In order to develop Goal Models for the available Viewpoints one must first identify each Viewpoint's take on the system's goals, that is, identify each viewpoint’s particular goals; this provides the analyst with a starting point for Goal Modeling. From the process of identifying the system viewpoints, one is left with the actions that caused the several viewpoints' identification. These actions portray, either in a generic or in a specific way, the types of interactions between these agents and the system and the overall objectives the system itself is supposed to accomplish. The objective at this stage is to identify the system goals in a localized manner, that is, contained in the viewpoint's scope, but however aiming at higher-level goals.

4.2.2.1 Produce List of Viewpoint Goals

Beginning with the several actions that served as a means to identify the system's viewpoints, one can observe those that refer to the system from a higher-level of abstraction or even those that although more specific, when merged describe a more generic system objective.

Looking at the first type of sentence:

"*In order to obtain a gizmo a user must supply his personal information and afterwards must validate the same gizmo by associating it to a bank account in an ATM."*

This sentence, although proposing several functional steps, ultimately translates into one objective for these two agents:

- Associate Gizmo with User.

However there is another type of sentence(s):

"*To access a parking lot, a user must place the gizmo in a readable place (most commonly the vehicle's windshield) and approach one of the entry machines*"

"*The machine recognizes the gizmo and, if it is a valid one, turns on a green light and allows the vehicle to enter.*"

In these, two differently timed actions are perpetrated by the agent or agents, but are too specific in their nature, quasi-requirements. Therefore, if one abstracts to a higher level of

those same actions, the merger of the two actions might translate into a high-level system-goal:

- Manage parking lot entry.

However, seeing as this should be done at a viewpoint's particular level, the first goal that was identified, should have been so while analyzing both the User viewpoint and the Gizmo viewpoint; they would share a common goal, but that discovery would be an outcome of an overview of all viewpoints.

Taking a look at the case study statement with these concepts in mind, we identify several key system goals for a viewpoint such as the User:

- Associate Gizmo with User
- Perpetrate Parking Lot Access
    - Perpetrate Entry
    - Perpetrate Exit
- Declare complaint

Or from the Gizmo:

- Associate with User
- Manage usage feedback
- Participate in Parking Lot Access

Having identified the particular system goals using the guidance of the already identified initial viewpoints, one still lacks the translation of these goals into specific requirements for each viewpoint: thus enters the KAOS Goal Models.

4.2.2.2 Produce a Goal Model for each System Goal in each Viewpoint

As it was previously referred, the initial set of system viewpoints will probably resemble the set of KAOS Agents that would participate in the Goal Models, therefore this step should integrate almost seamlessly with the previous constructions.

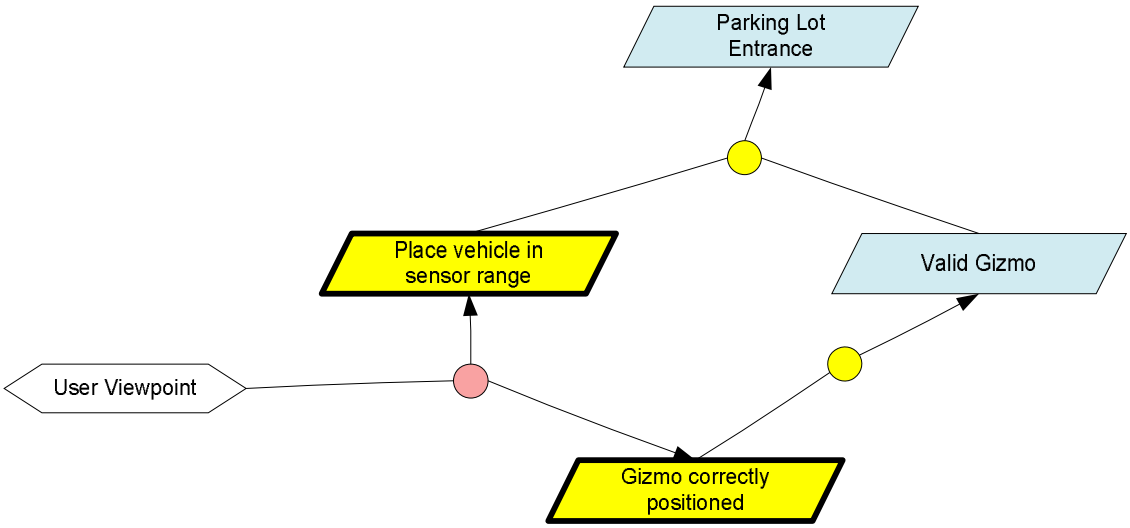

Figure 4-2 Parking Lot Entrance Goal Model for the User Viewpoint

Applying this step to the Goal of Perpetrating Parking Lot Entrances, this according to the viewpoint of system User, one would get the specific goal model regarding this particular subset of the system, as seen in Figure 4-2. It should be referred that barring any base document inconsistencies, if one would consider other viewpoints, for example the Entry Machine, one would add to the overall goal model of this particular Goal, but from a different point of view.

By adding this step to the Hybrid Approach Heuristics, one is introducing a clearly defined way of discovering the system's requirements, which was not apparent in the PREview approach, while maintaining a lightweight nature due to the graphical notation used for the goal decomposition process. Therefore, each viewpoint has a set of goals, but although these goals may be present in several Viewpoints as is natural, their decomposition need not be as such; by considering each system goal in the scope of each viewpoint it relates to, the goal decomposition process should only output the requirements and expectations that pertain specifically to the above viewpoint.

4.2.2.3 Obtain Set of Developed Viewpoints

It may be the case though that the initial set of viewpoints was too granulated or even, although that is unlikely due to the lightweight nature of their identification, too generic. In

the PREview approach the process of identifying viewpoints is an iterative one, although that is similar to almost every software development techniques, but at times these iterations may amount to too many, especially if the system viewpoints are particularly hard to identify and/or differentiate.

Taking into account the initial set of lightweight viewpoints and their goal models, the sorting out of mergers and separations, if based on these constructions, would be much more informed than it would be in the case of the PREview approach. Furthermore, it would be easier to clearly identify each agent's scope and thus those that overlap or are too broad.

This step would thus begin by reviewing the identified viewpoints and verify if, according to their goal models, any required a separation into more viewpoints, or if one or more viewpoints shared a common scope and would thus merit a merger.

Such would be the case, for example, of the Entry and Exit machines: if one were to study their particular goal models as shown in Figure 4-3 and Figure 4-4, one would verify that they would be too similar to be considered separately, but instead should merge into a single Viewpoint "Entry/Exit Machine", as shown in Figure 4-5.

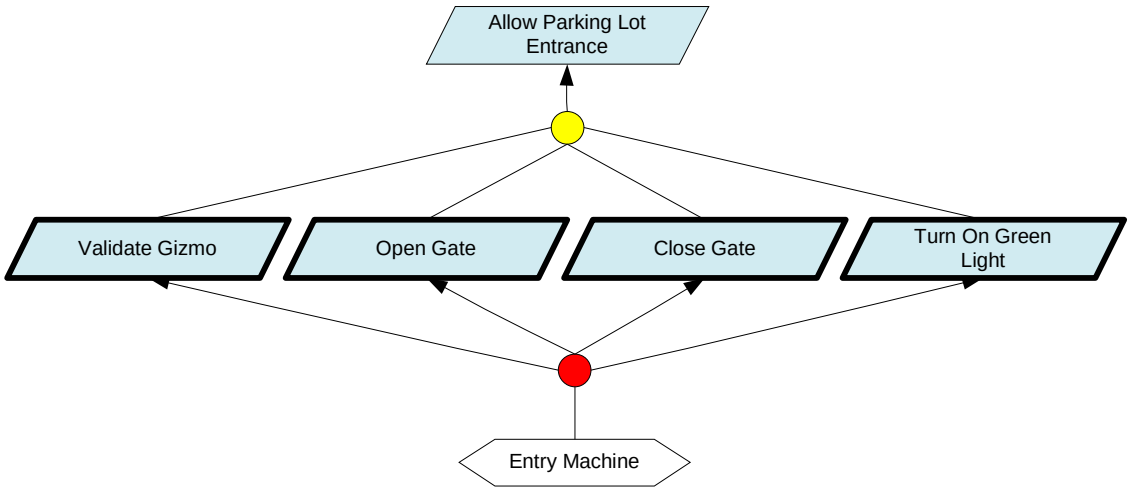

Figure 4-3 Allow Parking Lot Entrance Goal Model for the Entry Machine Viewpoint

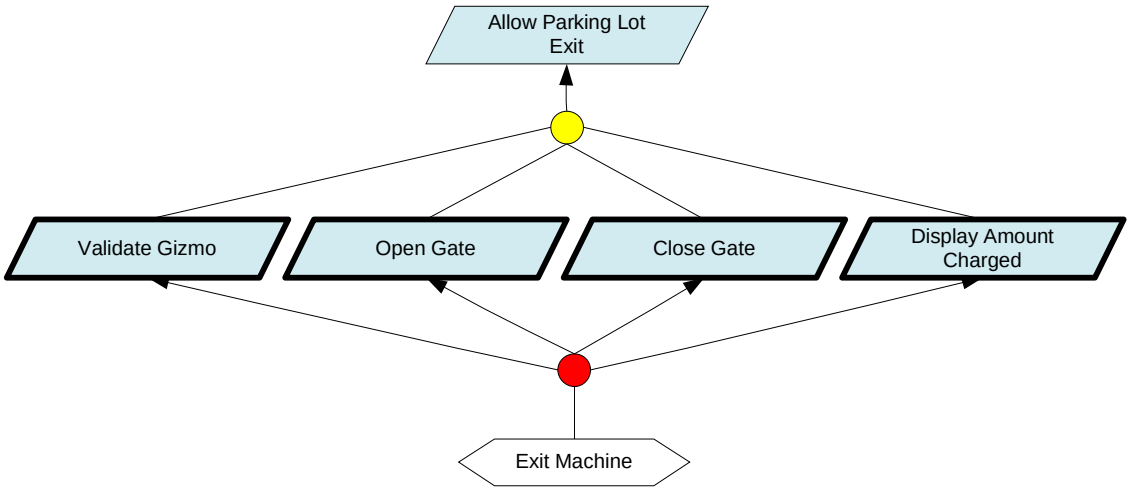

Figure 4-4 Allow Parking Lot Exit Goal Model for the Exit Machine Viewpoint

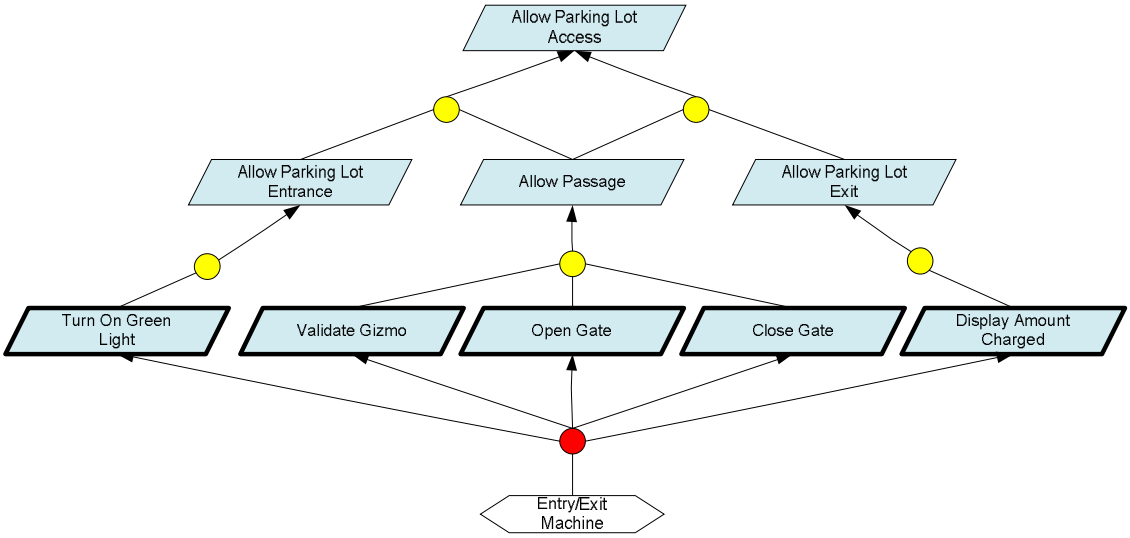

Figure 4-5 Allow Parking Lot Access Goal Model for the Entry/Exit Machine Viewpoint

Following this sifting through the set of initial viewpoints and contrary to the initial identification of the system's viewpoints, this particular iteration already takes into account the guidelines established by the PREview approach concerning the components obtained:

- The set of viewpoints should not be unmanageably large, both to reduce complexity and to maintain system perspective;
- the "final" set of viewpoints should be explicitly defined:
    - a **name**;

- its **type** (stakeholder or environmental);
- a **focus** – with more detail than the original viewpoints, using the goal models as guidance;
- **sources** – collected from the initial set of viewpoints and from goal model analysis and development;
- **history** – should detail if the viewpoint is a merger of several initial viewpoints, if so which ones, or if it is a sub-viewpoint of a larger one, and if so which.

Having the set of "final" viewpoints *a-la-PREview*, a more formal definition of these viewpoints is in order, both for structural and organizational purposes. This formal definition corresponds to the observation of the several Goal Models and the extraction of each viewpoint's requirements.

4.2.2.4 Elaborate each Viewpoint's Set of Requirements

Considering the several goal models that might be drawn for each viewpoint, the ensuing requirements and expectations can be withdrawn from the same models with some ease. However, if one is to look at the PREview requirements definitions, one can see that they have standard in what relates to syntax: actor A perpetrates action X; the requirements shown in the goal models though are less verbose and more "to the point", seeing as the graphical context does the explaining for them.

The requirements listing for a viewpoint is, however, essential for its clear definition. Therefore we are required to establish a method for withdrawing PREview requirements from KAOS Goal Models. Taking into account the previously described PREview textual representation of a requirement and its basic syntactic elements, we can find them while traversing the nodes of a goal model. Consider Figure 4-2 that represents a goal model for a User viewpoint; in this figure we can identify two expectations (which are the KAOS equivalent of PREview stakeholder viewpoints requirements):

- "Place vehicle in sensor range";
- "Gizmo correctly positioned".

As it was said, these requirements' textual definitions, when read like so are easily misunderstood, seeing as they are out of their graphical context. As such, a way to provide a better textual representation would be to include the graphical context in the said text: this can be achieved by traversing the Goal model's nodes from the requirement's agent to the top goal the requirement is decomposing, collecting the several textual representations as well as the relationship between the agent and the requirement, and registering the decomposition steps. Once again referring to Figure 4-2, these requirements would be translated into the following representations:

- "The User is expected to place the vehicle in sensor range in order to achieve parking lot entrance."
- "The User is expected to have the gizmo correctly positioned in order to achieve a valid gizmo and achieve parking lot entrance."

As we can see the user is expected to perform actions X and Y, seeing as in the models they are expectations; in the case it was a requirement, the user would be required to do perform X; furthermore, an expectation or a requirement X is always in order to achieve goal A and so on. This introduces the context for a requirement's description. More so, and in order to reduce verbosity in a viewpoint's requirements listing, requirements can be grouped by the goals they are meant to achieve:

- "In order to achieve parking lot entrance…
    - The user is expected to place the vehicle in sensor range
    - The user is expected to have the gizmo correctly positioned in order to achieve valid gizmo

Having a stable set of viewpoints (achieved in the previous step) and for each of them a clearly defined set of requirements, this information can and should be registered in a format that enables both quick consulting and intuitive organization: a tabular representation is therefore in order to maximize expressivity and organization.

Considering the viewpoint of the system User, taking into account the several goals it relates with, a table much like table 1 might come up.

Table 4-1 User Viewpoint Tabular Representation

Name	User
Type	Stakeholder
Focus	Regular user that interacts with the Via Verde system and may register and use Via Verde Gizmos in parking lots.
Requirements	1. In order to have a valid gizmo… a. The User is expected to purchase the gizmo b. The User is expected to activate the gizmo 2. In order to achieve parking lot entrance… a. The User is expected to place the vehicle in sensor range b. The User is expected to have the gizmo correctly positioned in order to achieve valid gizmo …….
Sources	Target test users of the application, city drivers.
History	-

4.2.3 Produce and Develop the System Concerns

Originally, the PREview approach would have begun with the identification of the system concerns, that is, the inherent traits every system functionality should respect. However, it is my belief that this step is less direct than what is desirable; too much is left to the analyst's capability to ask the right questions and too much is expected from the stakeholder's knowledge of the domain, which is more than often less than what would be required.

Therefore, seeing that defining a set of system concerns, although not necessarily the first, is still an important step in establishing the basis for requirement elicitation, these should be gathered from studying the system goals, that is, from each goal, and from their preliminary description, the analyst and the stakeholders should obtain a set of Primary-Concerns, or Softgoals, which, merged according to areas of interest, generate a list of system Concerns.

This of course is work that can be greatly enhanced by focusing each viewpoint in turn: although concerns are orthogonal in nature, their identification is based on information sources; what better sources then than the system’s viewpoints? While being encapsulating

units and thus more focused work environments, they still maintain a notion of the system in general by sharing common goals.

Furthermore, in order to minimize redundant work and seeing that concern sets should be of manageable size, it seems natural that Concern decomposition into external requirements can be achieved from the various sets of Softgoals that would be generated within each viewpoint's scope.

4.2.3.1 Produce List of System Concerns

Studying the case study's statement and the work achieved so far, especially relating to the identified viewpoints, if one would, for example, ask the Bank liaison to the project what concerns the institution would have regarding the system, Security would immediately be referred to as a main concern. Furthermore, if one would expose the system goals pertaining to that same Viewpoint to the same source, a list of specific concern requirements would emerge as properties of the system that should be safe-guarded.

On the other hand, if one would consult with the ATM or the Parking Lot Machine manufacturers, their main concern would obviously be Compatibility. They would in turn, when confronted with the list of system goals identified for the related Viewpoints, provide some insight as to where this compatibility would be most important.

When reviewing all of the Viewpoints' Concerns regarding the Via Verde system, one would probably come up with a list similar to the one that follows:

- Safety
- Multi-Access
- Response Time
- Availability
- Security
- Compatibility

This list would have been the product of collecting each viewpoint's concerns regarding their particular goals in the project.

This process of Concern discovery automatically establishes links between each Viewpoint and the Concerns it helped spawn, that is, the particular Softgoal (in this case Viewpoint-particular sub-Concern) that represents a higher level Concern according to that specific Viewpoint.

4.2.3.2 Develop Viewpoint Concern Requirements

At this point, another contribution of the KAOS approach is called forth: the use of Generic Goal Patterns. One of the fruits of the investment in KAOS technology has been the different systems it has been applied to and the similar patterns that have been possible to identify. These patterns are not a standard but more of a guideline, and are especially useful when dealing with more abstract goals as are the non-functional ones. By discovering requirement patterns in the identified concerns previous to their decomposition process, much of the analyst's work is reduced and one also benefits from the knowledge input of previous developers.

4.2.3.2.1 Verify Applicability of Generic Goal Patterns

If one would consider for example a generic pattern for a Secure system like the one in Figure 4-6 [21], instance the referred service with that of a Transaction, like in Figure 4-7 and applied it to the Security concern identified for the Banking Entity viewpoint, one would obtain a particular goal model as the one shown in Figure 4-8. The decomposition suggested by the generic pattern simplifies some of the work of the analyst by providing guidance to the ensuing process of eliciting external requirements.

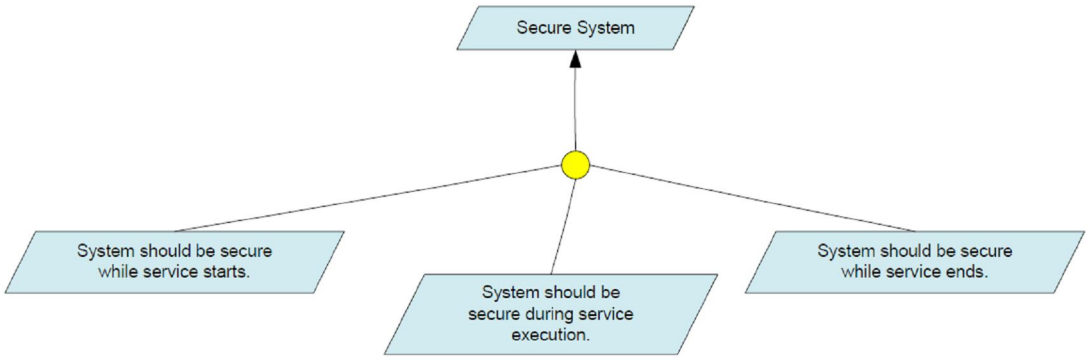

Figure 4-6 Secure System Generic Goal Pattern

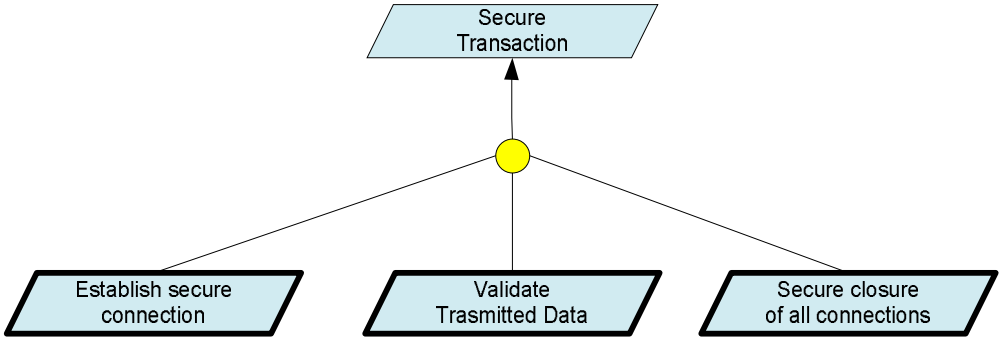

Figure 4-7 Secure Transaction Generic Goal Pattern

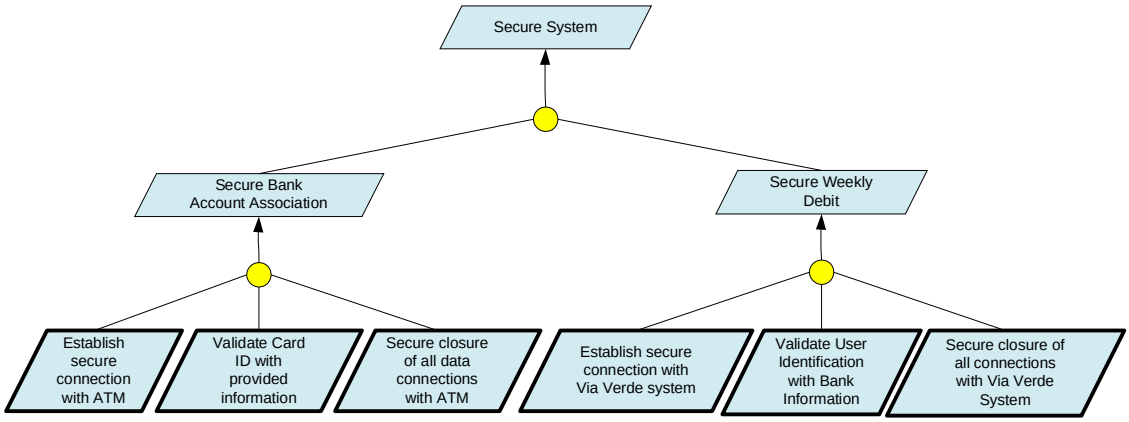

Figure 4-8 Security Concern Goal Model for the Banking Entity Viewpoint

4.2.3.2.2 Merge System-Wide Concern Goal Models

System concerns however are orthogonal in nature, and thus, although viewpoint-particular identification is helpful, their purpose is system-wide. Therefore, having obtained goal models for each concern in each viewpoint, one should compile them as seen system wide, that is, merge each System Concern sub-models.

These higher level models should not replace the particular ones, but instead be used to transmit to the analyst and to the stakeholders the overall concerns the system must take into account and their impact on its development.

This can be seen by once again considering the Security concern, but this time from a system-wide point-of-view: although it was not referred previously, the Security concern affects not only the Banking Entity viewpoint, but also the ATM and Entry/Exit Machine's Viewpoints; therefore a model as the one seen in Figure 4-9 would be produced, obtaining a system-wide view of the Security concern. The benefits one can withdraw from this change in the normal PREview process is that of using the goal models drawn for the several viewpoints to aid in the local identification of the concerns, whereas their identification would be simpler when with less focus but would still be able to be orthogonal after merging the several viewpoint-specific softgoals.

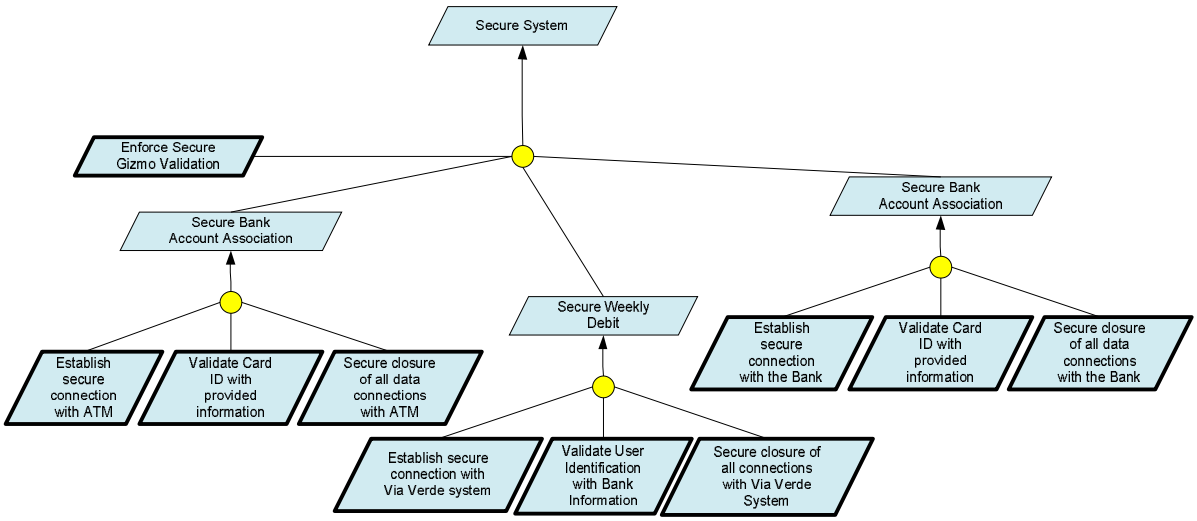

Figure 4-9 System-Wide Security Concern Goal Model

As we can see, the same “secure transaction” pattern is applied in the ATM viewpoint, and for the Entry/Exit Machine just the enforcing of the gizmo’s secure validation is required. It should also be referred that in the case of overlapping concern decompositions, they should also be merged into a single decomposition sub-tree, as would be the case of the ATM and Banking Entity’s “Secure Bank Account Association”.

4.2.3.3 Register Concern Requirements

This step consists of formalizing the developed goal models for each concern into a tabular representation similar to that of the viewpoints. This representation is similar to that of the PREview approach, both creating a separate representation for each concern as well as adding particular references to the tables already produced for each viewpoint.

Table 4-2 represents the Security concern as viewed system wide, using the textual template already utilized when describing viewpoint requirements.

The identified requirements would then be added to the respective viewpoints’ tables.

Table 4-2 Security Concern Tabular Representation

Concern	Security
Affected Viewpoints	Gizmo, Banking Entity, ATM
Requirements	1. In order to achieve a secure system… a. The system should enforce gizmo validation. b. The system should achieve a secure bank account association and in order to do so… i. The system should establish secure connection between Banking entity and ATM. ii. The system should validate card ID with provided information. iii. The system should secure closure of all data connections between bank and ATM. c. The system should achieve a secure weekly debit and in order to do so … i. The system should establish a secure connection with the bank. ii. The system should validate user identification with bank information. iii. The system should secure closure of all data connections with the bank.

4.2.4 Represent the System's Obstacles and their Solutions

Another contribution introduced by the KAOS approach is the explicit and differentiated declaration of the system obstacles. It would be naive to think that every system goal is accomplished cleanly and without missteps, thus the PREview approach included in the description of each viewpoint's requirements what could be called as "conditional" requirements, something similar to "if scenario B does not occur then". However, this

identification of the system's obstacles was tightly coupled with the requirements it was related to, and might even imply redundant developments when describing these negative scenarios.

The KAOS approach and the Objectiver methodology solve this problem by introducing the graphical representation of an obstacle in a Goal Model; each Goal may or may not be linked to an exceptional scenario called Obstacle that would impair the completion of that Goal. Directly related to this concept is also the concept of a Solution, that is, another system Goal that is derived from the need to mitigate a certain Obstacle. These components represent deviations of the normal course of events, "worse-case" situations that must be contemplated when designing a system and whose solution must be identified. Furthermore, due to the fact that the Hybrid Approach uses Viewpoints as encapsulating units, the identification of such obstacles can be done in a more localized fashion.

In KAOS, this step would have been introduced into the process right before the elaboration of the several Goal Models, that is, in possession of the basic system goals, functional and non-functional, the analyst would immediately begin to consider non-optimistic scenarios. However, it is only when the analyst begins to decompose the several goals that the obstacles become clear and their implications translated into explicit situations; furthermore, non-functional goal models may themselves propose specific obstacles to functional goals, thus the inclusion of this step only at this stage of the process.

This fact is evidenced when considering the previously referred issue of a Gizmo Validation: from the early stages of studying the problem it is clear that issues of lack of validation should be addressed, however, it is only when one decomposes the goal of parking lot access that one clearly identifies the moment in which that lack of validation is made clear, and its solution as well.

Contemplating the issue of gizmo validation from the Entry Machine Viewpoint as seen in Figure 4-10, an obvious obstacle to a valid gizmo is the fact that the same gizmo may not have been activated. Like in the original KAOS approach, that obstacle is represented, as well as the machine's response to that event: an orange light is turned on.

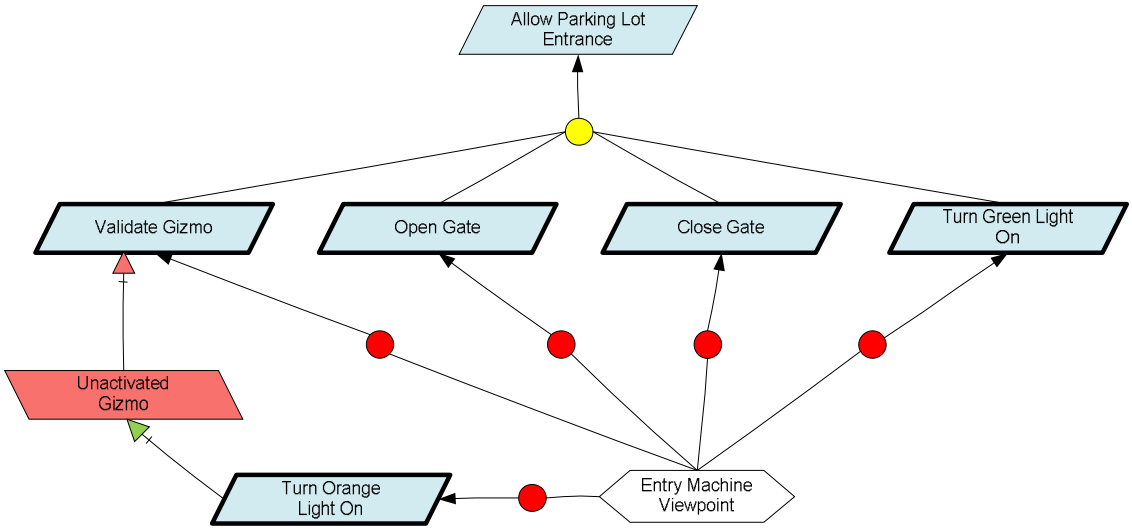

Figure 4-10 Allow Parking Lot Entrance Goal Model for the Entry Machine Viewpoint with Obstacles

Another Goal Model, in this case regarding the Banking Entity Viewpoint, can be drawn regarding the goal of Realizing Weekly Debits: the Banking Entity is responsible for charging the weekly total to the user's account and if any error occurs (an obstacle) the Main System is immediately notified and deals with further notifications; this goal model is shown in Figure 4-11.

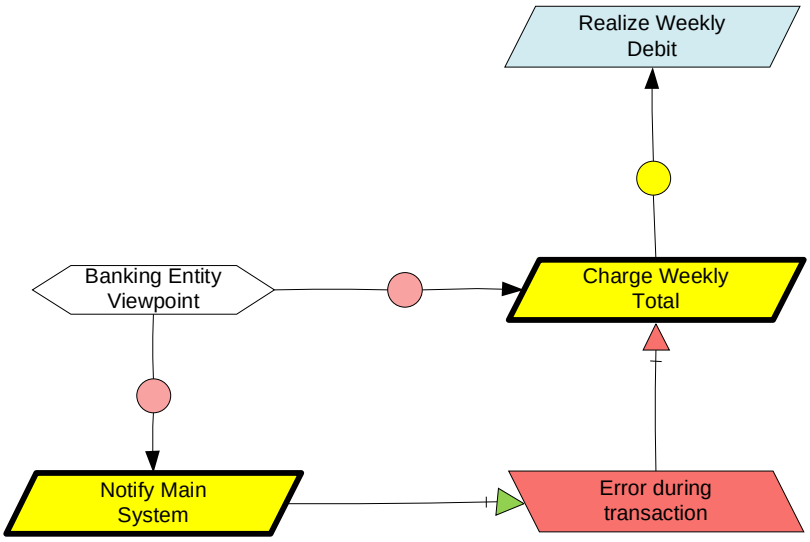

Figure 4-11 Realize Weekly Debit Goal Model for the Banking Entity Viewpoint with Obstacles

To this point there is nothing new regarding the Hybrid approach's treatment of obstacles. However, when transposing these obstacles to the viewpoint's description in the tabular representation, a small change is introduced: when dealing with a viewpoint's requirements, events might take different paths for two reasons, either there are simply optional successions of events or there is a main succession of events that may take an exceptional turn; this exception is what is considered an obstacle. PREview treated both types of situations the same way, as "if-then-else" descriptions; however it is my opinion that if the KAOS-detected obstacles were to translate into textual additions into the viewpoint description as it can be seen in Table 4-3, the mere distinction might be an important contribution to the Viewpoint's description.

Table 4-3 Entry/Exit Machine Viewpoint Tabular Representation

Name	Entry/Exit Machine (EEM)	
Type	Environmental	
Focus	Machines positioned at parking lot entrances and exits to control gizmo validation and vehicle passage.	
Requirements	1. In order to allow parking lot entrance… a. The EEM is required to validate gizmo b. The EEM is required to open gate c. The EEM is required to close gate d. The EEM is required to turn on green light.	
Concern Requirements	…	
Obstacles	Opposed Req.	1.a
	Obstacle	Inactivated Gizmo
	Solution	S.1.a. The Entry Machine is required to turn on orange light
Sources	Machine manufacturer, parking lot frequent users.	
History	Previously Entry Machine and Exit Machine Viewpoints.	

It should be referred though, that for comparative analysis purposes, solutions to the identified obstacles should be treated as viewpoint requirements.

4.2.5 Requirements Analysis

According to the PREview approach this would be the perfect moment to enter the stage of Requirement Analysis: our Knowledge Base (KB) is sufficient and therefore should be analyzed to guarantee completeness and correctness. This analysis would be done in different steps organized according to the impact each of them would have on the KB.

4.2.5.1 Perform Inter-Viewpoint Interaction Analysis

First of all, there are several key concepts to consider regarding the PREview approach:

- Viewpoints are encapsulation units and as such, one of the benefits of using them is the ability for localized and independent development;
- Due to their independent nature, it is highly likely that when eliciting different viewpoints conflicting requirements might emerge, even more so if each viewpoint is treated by a different analyst;
- Viewpoint requirements are *functional* and therefore, when conflicts emerge between them, they cannot be ordered according to importance:
    - Ex: if, on the one hand, a car must pass through the gate, and on the other, it mustn't, a conflict emerges that cannot be solved by attributing relevance.

Having these concepts in mind, it is necessary to determine the degree of interaction between every Viewpoint's requirements, that is, for each Viewpoint Requirement, identify if there are other independent, conflicting or overlapping viewpoint requirements (this issue may have already been inadvertently addressed while compiling the system's initial viewpoints into the "final" set). This is done by means of an Interaction Matrix: on each axis of the matrix a Viewpoint's requirements are listed; the requirements that generate conflicts are highlighted (the cell in which they meet) and must be discussed according to their status exactly like in the PREview approach.

It should be emphasized that conflicts discovered during this type of analysis cannot be solved: conflicting viewpoint requirements are merely the fruit of independent requirement elicitation and must be treated as "faulty elicitation".

Relating to the case study, the inclusion of the two-step requirements identification allowed for a more finely grained definition of viewpoints, which caused the only two viewpoints that were likely to present overlapping requirements to merge: Entry Machine and Exit Machine. As seen in the section where that merger is referred, both of these viewpoints required validations to occur and gates to open and close, the ensuing table would reflect precisely that characteristic.

4.2.5.2 Perform Inter-Concern Interaction Analysis

In what relates to same type of analysis between Concerns, the fundamental concepts are different:

- Concerns are orthogonal in nature, and as such, their requirements are non-functional, that is, they do not describe specific actions, but instead specific qualities the system must possess;
- Qualities in turn have a wider scope of impact on the system and can be ordered in terms of importance:
    - Ex: The system should be quick to respond, but it also should be secure.

This of course can be sorted out in terms of what is more important for the stakeholders.

For these reasons, inter-Concern analysis, as taken from the AORE approach [20] can be done at its highest level: an interaction matrix would be produced with a list of the system concerns on each axis, and their interaction would be studied according to the effect they have on each other, that is, it they do not affect each other, if they are beneficial or if they have a detrimental effect (Ex.: Security almost always has a negative impact on the system's Response Time).

However, seeing as we are focusing on the advantages of the KAOS models, identified conflicts should be reflected in a system and concern wide model, displaying the merged models of all concerns, and signaling any conflicts with the correspondent KAOS notation.

That notation would then be extended to include not only conflicts but also cooperation, adding new notation symbols to represent the cooperation relationship as well as information regarding the direction of that influence ("-" for negative contribution, or conflict, and "+" for positive contribution, or cooperation).

However, a note to consider is the necessity for a possibly complex model as it would be the overall concern model: would the necessary overhead that such a model would create justify the outcome? We believe it would not. However, there is still an advantage in considering the specific impact of each concern in each viewpoint when considering an overall model. Therefore, it was decided that when merging all concern models, the overall model would limit each sub-model to the viewpoint specific levels, that is, when each concern model would begin to specify its impact on each viewpoint.

Applying this methodology to the Case Study, one would obtain a model as shown in Figure 4-12, in which the several system concerns are analyzed for effects produced on each other.

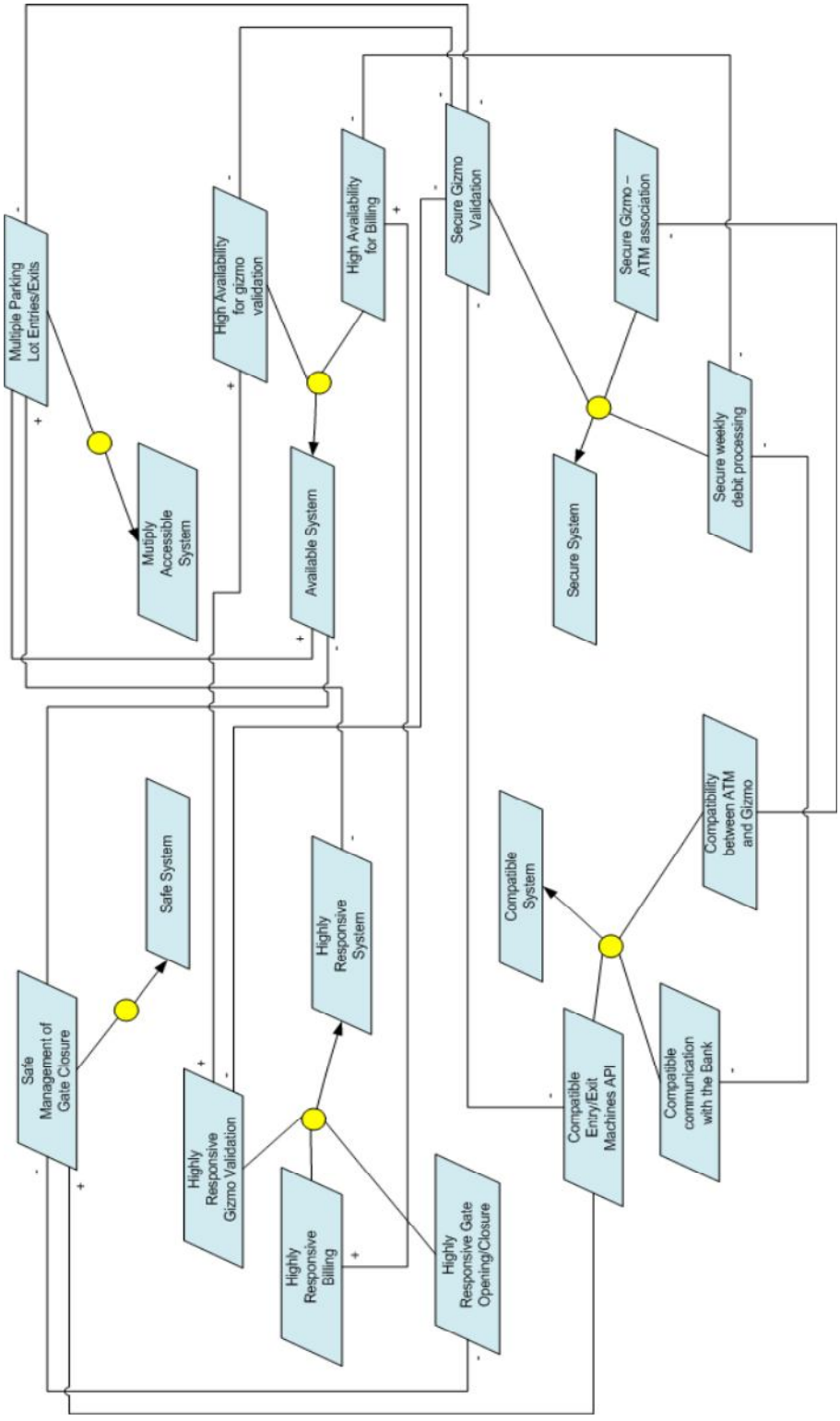

Figure 4-12 System-Wide Concern Goal Model with Comparative Analysis

4.2.5.3 Perform Viewpoint-Concern Interaction Analysis

Afterwards, and borrowing from another contribution from the AORE approach [20], there is a need to study the effect of the several system concerns on the different system viewpoints: on one side of the matrix a list of Viewpoints, on the other a list of Concerns; going back to step 2 this matrix is easily filled by analyzing the connections between viewpoints and their softgoals, both in the Viewpoints and the Concerns' tabular representations.

It should be referred though, that this second matrix does not indicate conflicts or redundancies, but instead merely summarizes the contributions of each Concern to each Viewpoint. The application of this step to the case study is shown in Table 4-4.

Table 4-4 Viewpoint-Concern Interaction Analysis

	User	Employee	Bank	Machine	Gizmo	Accounting	Vehicle	ATM
Response Time	X	X		X	X		X	X
Availability	X	X		X	X			X
Security	X		X			X		X
Compatibility				X				X
Safety	X						X	
Multi-Access	X			X	X		X	X

4.2.6 Requirements Negotiation

Finally, the PREview approach would end its process by entering the Requirements Negotiation stage. Normally, this would be a stage that would have as a basis all the matrixes produced in the analysis stage and that would generate input for both the several viewpoints' and concerns' requirements. In an attempt to simplify the negotiation stage, another concept is borrowed from the AORE approach: the Weighted Contributions Table.

4.2.6.1 Produce a Weighted Contributions Table

This table is a merger between the results of the previous two steps: a matrix exactly like the one in Table 4-4 would be produced and for each Viewpoint column or row, the several highlighted Concerns would be analyzed according to the model in Figure 4-12; if a negative contribution exists between two or more of those concerns it is required to define their relative importance, that is, when developing Viewpoint X, Concern A would come first, then B, and so on; this would be done by attributing weights to each conflicting concern, on a percentile scale.

This type of analysis can only be done with the presence of the system's stakeholders, since they are the only ones that can decide which concerns they consider more important. Therefore, by producing Table 4-5, one is satisfying the requirements of the Negotiation Stage: deciding on a stable set of priorities for subsequent changes.

Table 4-5 Weighted Contributions Analysis

	User	Employee	Bank	Machine	Gizmo	Accounting	Vehicle	ATM
Response Time	1.0	X		X	1.0		1.0	0.9
Availability	0.8	X		X	X			0.9
Security	0.9		X			X		1.0
Compatibility				1.0				0.7
Safety	0.9						0.9	
Multi-Access	0.8			0.9	0.8		0.8	0.8

Taking a look at Table 4-5 applied to the case study, one can see that for example, when dealing with the User Viewpoint, if one is to take a look at Figure 4-12 and Table 4-4, one verifies that this viewpoint considers both the Response Time and the Security concerns, however they are conflicting in nature, that is, when considering one of them the other is normally relaxed. Therefore stakeholders are required to define priorities for these concerns regarding this specific viewpoint, in this case, for the User it is perhaps more important that the system responds as quickly as possible.

The KAOS approach would follow onto more detailed stages of development, namely the Operationalisation of each Requirement. However, the PREview approach limits itself to the Requirements Management process, which is one of the attributes that makes it such a lightweight approach. In the spirit of maintaining a certain degree of that lightweight nature despite all the aspects that were introduced by the KAOS approach and even those borrowed from the AORE approach, it was decided that this Hybrid Approach too should limit itself to these three stages of requirements elicitation: Discovery, Analysis and Negotiation.

4.3 Summary

The Hybrid Approach, whose heuristics were previously described, constitutes an extension to the original PREview approach [25], complemented by components of the KAOS approach [13] and inspired by the conflict resolution techniques of the AORE approach [20].

The procedures we describe in this chapter therefore differ slightly from the original PREview process, seeing as some concessions had to be made in order to allow as seamless an integration as possible regarding the components from the other approaches.

The process of this extended PREview approach we have dubbed Hybrid Approach is shown in Figure 4-13. The generic intentions of PREview are still apparent in the three main stages of the approach: Requirements Discovery, Analysis and Negotiation; however there were some elements that in their introduction triggered modifications in the original process.

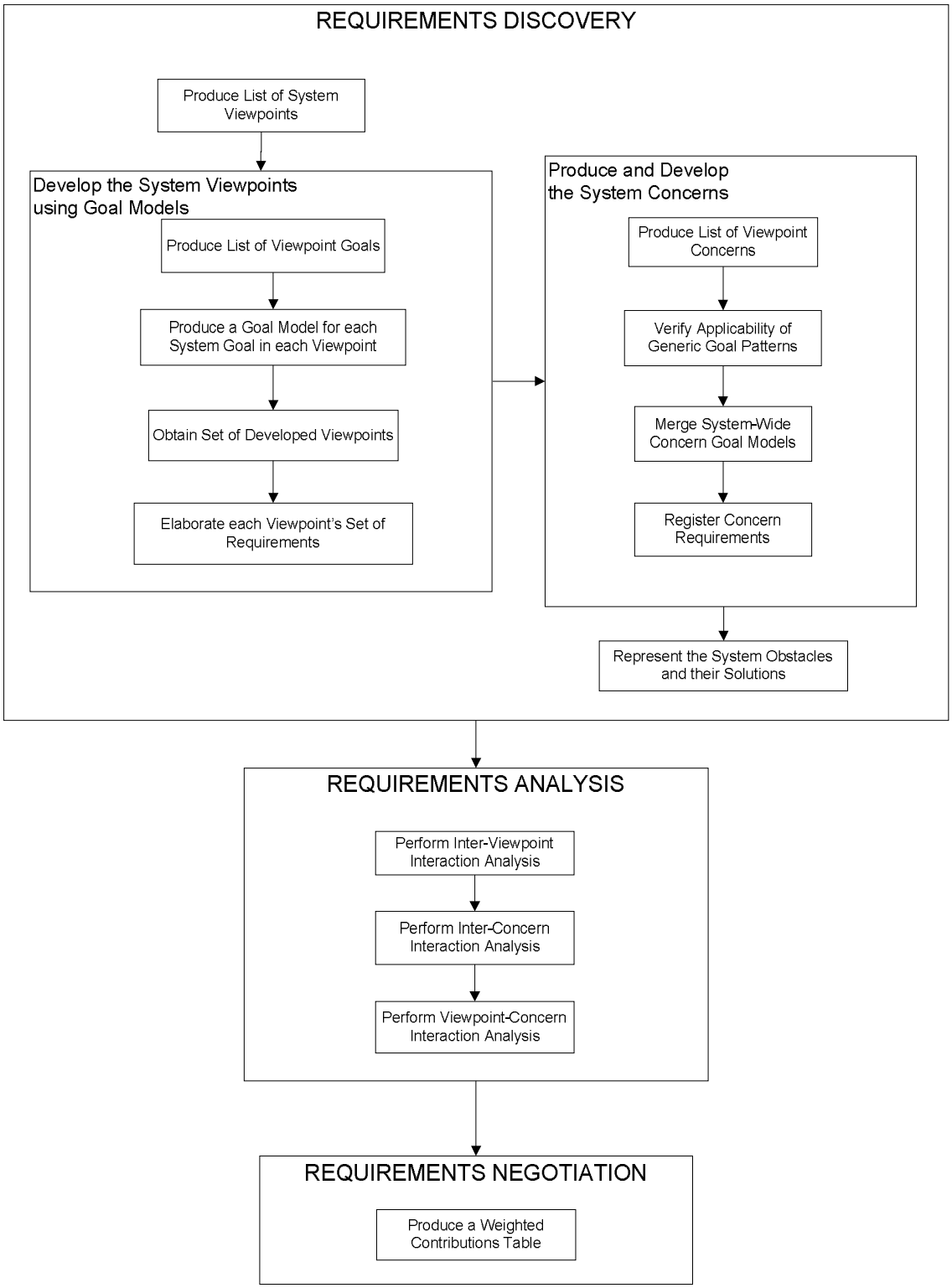

Figure 4-13 Hybrid Approach Process Model

By removing the concern discovery process from the early stages and introducing the notion of initial viewpoints applied to the development of KAOS goal models, this Hybrid Approach attempted to mitigate the lack of a clearly defined requirements discovery mechanism in the

PREview approach. As was seen by the case study chosen to describe that process, by following the introduced steps an analyst can easily arrive on a stable set of viewpoint requirements and, if need be, clarify the viewpoints' definitions supported by the requirements they entail.

The lack of an existing set of concerns registered no impact on the discovery process and furthermore, the process of obtaining those same concerns was greatly enhanced by the existence of the several goal models and the identification of each of them in smaller scopes as were the viewpoints.

The inclusion of the KAOS notion of obstacle and providing their solutions aimed at clarifying another step of the PREview approach that was the existence of conditional requirements. As such, erroneous situations and system options are described in a highlighted manner in order to clearly evidence them.

Requirement Analysis in PREview was also a subjective process and by introducing components of the AORE approach, this work intended to clarify the conflict identification and solution procedures. Complementing these AORE elements, the use of the KAOS graphical methodology, extended by additional symbols, aimed at providing the analyst the required tools for an easy identification of concern conflict and/or cooperation.

Finally, the negotiation stage in PREview was also a subjective stage, relying on the analyst to ask exactly the right questions with possibly little knowledge of the system's issues. This approach attempted to provide an additional aid to the analyst by adding a weighted contributions table displaying conflicting concerns and offering the stakeholders a simple method of registering their opinions regarding potential priorities.

This approach has a documental weight that is far superior to the PREview approach, however, we believe that this additional overhead is required to minimize analyst errors and maximize the requirement elicitation process. This fact is further explored in the following chapter by applying the previously described heuristics to an industry case study.

Chapter 5
The Br@in Case Study

This chapter describes another case study called Br@in, a real-world industry telecommunications billing application, with the intent of demonstrating the application of the previously defined heuristics of the Hybrid Approach.

5.1 About the Product

Br@in (Billing Resources @ IP Networks) was born as a software platform for the automatic treatment of billing operations for an organization. However, throughout the course of its lifetime and based on previous successful modular implementations (specific add-ons for different clients), Br@in's major goal has evolved into controlling all costs of an organization's human resources.

These include not just telecommunications (like previous versions of the product), but also costs related to cell phone usage, laptops, room bookings and many other assets that may be configured with Br@in.

The product must encompass two main components: the management of all the information required for the billing process and the interaction with the actual Br@in users.

For the first component, data analysis, Br@in is required to integrate specific connectors for several technologies, allowing to concentrate the billing of a large amount of services onto a single entry point:

- IP Telephony;
- Traditional IPBX Telephony;

- GSM Telephony;
- Data Transference (Wan and Internet);
- Video Conferences

Data must be imported from these several services, it must be analysed and unified in order to generate a common source of information for the subsequent processing and generating of a variety of cost reports according to pre-established criteria. These processes should be scheduled to occur at specific times.

Regarding the second component, user interaction, Br@in must provide a carefully studied interface that differentiates users according to their roles in the organization, enabling them to perform role-according tasks: a normal User will be able to visualize system output according to the data that stems from the information provided by the telephonic centrals, and an administrator will be able to manage the several entities that make up the Database produced by the same process.

5.2 Context Information

Seeing as Br@in must be able to control all costs of an organization's human resources, several concepts of a generic organizational structure must be made clear, both from the human resources point of view and from the services' components that must be billed:

- A user is any human resource of an organization.
- An extension identifies a phone line used for making calls and may be associated with users.
- Departments are all organizational nodes present in the organization's structure and may be organized according to a hierarchy.

- Project accounts are special structures that allow enveloping cost centres in the Br@in solution. An account may be associated with one or more departments according to a particular percentage of the department's involvement in the project at stake.

- A user can belong to one or more departments, also taking into account the percentage of the user's work for each department.

- A plafond is a pre-defined limit for call costs attributed to each user.

- GSM Devices are all the cellular phone numbers the organization makes available for its collaborators.

- Assets are all organizational possessions that may be associated with users and thus be the target of billing:

    - Ex.: Renting of rooms for meetings, cellular phones, laptops, desks, square footings occupied by a particular office, etc

- Like other services, asset billing may also be associated with particular Departments, Project Accounts or Users.

Although Br@in's scope is expanded from its original goal, telephony billing is still its main purpose, and therefore there are several other important concepts that should be understood, regarding this particular field:

- IPT/PBX Calls are all the calls made or received through the organization's extensions. These call records, when taken from the organization's central telephony processing unit, can be classified according to their success or lack thereof and according to their origin and destination (inbound, outbound or internal).

- An operator is the contracted agent that enables the organization's telephony capabilities.

- A gateway is a physical exit or entry point of a call into an organization's telephonic network.

- Tariff Plans are the several billing options made available by an operator to its clients.

- Prefixes identify the operator and location of the destination of each call. They can be associated with cost ranges according to specific tariff plans.

Dealing with telephony communication centres, Br@in should be able to gather call related information from their databases, with a particular emphasis on allowing for different types of data sources (Cisco, Avaya, Ericsson, etc).

This data collection should be done in the background as to minimize impact on the user experience (Call Detail Records (CDRs) for a medium sized company are in the thousands per year, therefore there is much data to collect, unify and process).

5.3 Br@in App

Based on the data collected, the application itself should provide Br@in users with a means to access that information and manage it according to their privileges.

A distinction should be made at least between Administrators and simple Users, granted that Administrators must have full access to the application, namely configurations and output, whereas Users may only have access to the output of information.

In what relates to the Administrator, there are several tasks that it must be able to perform:

- Consult the status of the Data Collection processes that may be occurring, change their periodicity and enforce their execution;
- Consult tracing information regarding the Data Collection processes, visualizing logs in a very interactive manner (sorting criteria);
- Manage the sending of reports by email, allowing for specific scheduling of automatic report sending for different destinations and/or groups, as well as the addition of destinations;
- Control the size of the several Databases used for storing data from the background processes, allowing for archiving old information or deleting it;

- Visualize several statistic information regarding the database stored information from the data analysis processes;
- Manage the several Telephonic Centrals that are associated with the system, allowing to add new ones and edit or eliminate existing ones;
- Handle the several system gateway locations, as well as the gateways themselves;
- Manage the system's telephony operators, their tariff plans and device prefixes;
- Visualize and re-organize the hierarchical structure of the organization, allowing to add, edit or remove any of the hierarchical entities referred in previous sections (users, departments, project accounts, etc) as well as any associations that may exist between them and billable components (extensions, assets, etc);
- Asset management, with particular emphasis on the definition of asset types and the costs they entail, as well as their association with hierarchical entities;
- Importing of all types of information regarding the system's database (users, devices, tariff plans, etc) from other formats (ex: Excel).

The plain Br@in User should only be able to visualize output from the application:

- Call details (origin, destination, duration, etc);
- Call (IPT/PBX and GSM) and asset costs;
- Reports according to pre-defined parameters;
- Export to other formats (graphics, xls, etc).

Of course, output should also be available to Administrators.

Br@in should also be able to present alerts to the users, either through the application itself or by sending emails; these alerts might relate to any system anomaly the user's privileges allow him to be aware of.

5.4 Br@in and the Hybrid Approach

The developed hybrid approach comprises, as was previously defined, of 6 major steps, each of them divided into their own particular stages:

1. Produce List of System Viewpoints
2. Develop the System Viewpoints using Goal Models
    a. Produce List of Viewpoint Goals
    b. Produce a Goal Model for each System Goal in each Viewpoint
    c. Obtain set of developed Viewpoints
    d. Elaborate each Viewpoint's set of Requirements
3. Produce and develop the System Concerns
    a. Produce List of Viewpoint Concerns
    b. Develop Viewpoint Concerns Requirements
        i. Verify applicability of Generic Goal Patterns
        ii. Merge system-wide Concern Goal Models
        iii. Register Concern Requirements
4. Represent the System's Obstacles and their Solutions
5. Requirement Analysis
    a. Perform Inter-Viewpoint interaction analysis
    b. Perform Inter-Concern interaction analysis
    c. Perform Viewpoint-Concern interaction analysis
6. Requirements Negotiation
    a. Produce a Weighted Contributions Table

As can be read in this chapter’s previous sections, the Br@in case study was chosen as a more complex subject for applying this approach. After demonstrating the approach’s steps and the benefits withdrawn both from each step in separate and from their succession by using the Via Verde case study, the goal with this more complex project statement is to demonstrate the described advantages in a clear and unequivocal manner and with a greater scope. Furthermore, seeing as the Br@in project is integrated in a real industry context, the conclusions that may be withdrawn from the hybrid approach’s application have a far more credible significance.

5.4.1 Step-by-Step

5.4.1.1 Produce List of System Viewpoints

The first step of the approach consists of a loose identification of the project's viewpoints; it is called a "loose" identification because of two main factors, as explained previously and now revised:

- First of all, to differentiate the viewpoints that come as output from this process from the PREview viewpoints;
- Second, because loosely defined viewpoints are closer in concept to the KAOS agents, and thus easier to integrate with the ensuing Goal Models.

Although these initial viewpoints can be obtained in many manners, the hybrid approach suggests using both their separation into two categories: stakeholder related viewpoints and environment related ones; as well as the tactic of syntactic analysis: verbs in sentences portray the actions that define the project's objectives and their actors or objects provide insight into what Viewpoints should be regarded.

Applying this tactic to the Br@in project statement there are several sentences that emerge immediately:

"A normal User will be able to visualize system output according to the data that stems from the information provided by the telephonic centrals"

From this particular sentence two different actions stand out: visualizing system output and data stemming from the telephonic centrals. These different actions are in turn perpetrated by two different agents: a normal User and a set of telecommunications centrals. Each of these agents is of a different type, seeing as a User is stakeholder related and the telephonic central is part of the system environment.

"An administrator will be able to manage the several entities that make up the Database"

In this sentence one can again easily identify a stakeholder related viewpoint, the Administrator, by associating it with managing the several entities that are part of the system database. This management capability is detailed in the Br@in App section, where it is said for example that an administrator can:

"Consult the status of the several Data Collection processes that may be occurring, change their periodicity and enforce their execution"

or

"Manage the system's telephony operators, their tariff plans and device prefixes"

If one would extend this reasoning to the remaining project statement, the resulting list of the system's initial viewpoints would be as follows:

- **Environmental:**
  - Telephonic Central - software or hardware components that handle telephony processing in the company and provide call information to the system
- **Stakeholder:**
  - User – regular utilization of the application – output visualization with minimal configurations
  - Administrator - management every aspect of the application and the information it handles.

These viewpoints were precisely the ones detected in the previous examples, which merely reflects the fact that the Br@in system is mostly reliant on itself, involving a minimum of foreign components and/or intervening agents; such was not the case, for example, of the Via Verde case study.

The tabular representation of the system's viewpoints at this stage is important as it always is, despite their temporary nature: it is an organized method for exposing the acquired information and may ease the possible merger and separation processes by allowing manageable referencing of initial viewpoints instead of explicit naming.

However, as was explained in the approach's heuristics, this representation, as the Viewpoints it describes, is a loose one, and aims also at reducing the time spent in documenting the process. Examples are shown in Table 5-1, Table 5-2 and Table 5-3.

Table 5-1 Administrator Initial Viewpoint

Name	Administrator
Type	Stakeholder
Focus	The top-ranked user of the application. Manages every aspect of the application and the information it handles.

Table 5-2 User Initial Viewpoint

Name	User
Type	Stakeholder
Focus	The regular user of the application. Visualize output with minimal configuration possibilities.

Table 5-3 Telephonic Central Initial Viewpoint

Name	Telephonic Central
Type	Environmental
Focus	Software or hardware components that handle telephony processing in the company and provide call information to the system.

In what concerns stakeholder viewpoints, it is hard to "escape" the entities they relate to and thus it is highly likely that initial viewpoints may not change entering the definition stage, but when it comes to environmental agents, focusing on particular needs, besides facilitating the identification process, may benefit the elaboration of the following Goal Models, while not harming the encapsulation advantages that come from using viewpoints.

5.4.1.2 Develop the System Viewpoints using Goal Models

As was stated in the heuristics chapter of this thesis, one of this hybrid approach's most important steps is one taken from the KAOS/Objectiver method: Goal Modeling. This step translates a clearly defined heuristic approach at requirements decomposition into a set of graphical components that intertwine into easily understandable models of a system's goal-into-task decomposition.

Furthermore, this hybrid approach extends that modeling capability with the encapsulation advantages of the PREview approach's concept of viewpoint, thus mitigating one of the few and main issues of the referred models: the fact that KAOS models, as has been said in this work, have the tendency to become complicated webs of intertwined entities when the project surpasses a certain degree of complexity.

This although should be done in a single step and thus the first thing the analyst should do is to regard each individual viewpoint and consider the system goals that led to its identification.

5.4.1.2.1 Produce List of Viewpoint Goals

Applying this step to the Br@in case study, although the considered viewpoints may possess more identified goals, the ones that are referred are those are both more diverse in their modeling as well as those that are more interesting to represent in a Goal Model, seeing as demonstrating all of them would become redundant.

The initial environmental viewpoint known as Telephonic Central is responsible for producing the database information, specifically by forwarding the raw call detail data to the system:

- Telephonic Central – Provide database information - provide raw call detail data.

Focusing a stakeholder related viewpoint, the application User is responsible for, among other things, consulting call and asset cost details, as well as “ordering” system reports. These are obviously UI oriented objectives, but alas so is the basis of the User’s participation in the project’s requirements:

- User – consult output cost details, produce reports.

And finally, the system administrator is responsible for a great deal of management objectives, both regarding database entities as well as configuring other system components. We will focus on three particular goals for ensuing modeling:

- Administrator – manage hierarchical structure, manage reports automation and manage assets.

The goals these three viewpoints present and that compose the subset that is considered in this stage are representative of the different nuances both stakeholder and environmental viewpoints might demonstrate: the User viewpoint’s goals define it as an agent of simple intended interaction with the system, a user visualizes output and specifies reports with minimal configuration; the Administrator viewpoint’s goals however portray it has stakeholder viewpoint that interacts with the system in a more complex way, that is, besides the basic user interaction, the administrator configures several system modules and performs many database related tasks. On the other hand, the environmental viewpoint that was identified as Telephonic Central forwards information to the system on demand.

5.4.1.2.2 Produce a Goal Model for each System Goal in each Viewpoint

This stage of the hybrid approach’s process is when the system’s requirements will start to take form and where most of the approach’s advantages are evident. Like it was said in the thesis’s motivation, PREview’s approach to requirement elicitation leaves too much room for analyst subjectivity to step in; this deems it an approach that perhaps depends too much on the analyst’s ability and/or experience. One of the KAOS approach’s main advantages however, is its formalism and its insistent and clear definition of each and every step of the requirement elicitation process; furthermore, the mechanism known as Goal Decomposition assures that

beyond obtaining a goal's requirements, the goal modeling itself will be complete, as every goal must correspond to a requirement or expectation.

The previous step in this approach already mitigates some of the PREview missing pieces: by providing system goals to each viewpoint, it is a first step at obtaining its requirements; furthermore, since goals are more generic in nature, they are more clearly identified.

Looking at the Telephonic Central's viewpoint and its goal, one can see that there is a top goal for the viewpoint, which is to Produce Database Information. "How will the Telephonic Central achieve that?" is the question that the goal decomposition mechanism imposes while focusing on this particular viewpoint; by reviewing the project statement and considering the instancing of this goal, the forwarding of call detail information is a specific task, and thus a requirement of the top goal.

This however is a textual description of a process that, being undertaken in a graphical way, produces a model like the one in Figure 5-1. As one can see, the top goal is, as was stated, "Produce Database Information" and is decomposed into the "Provide Call Detail Records" requirement, directly linked to the Telephonic Central agent.

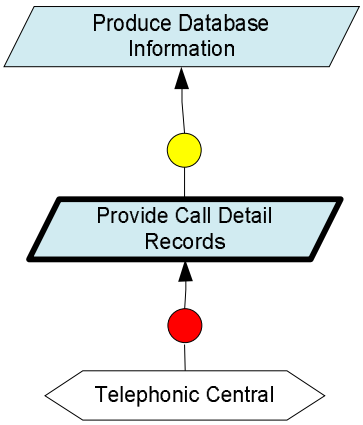

Figure 5-1 Produce Database Information Goal Model for the Telephonic Central Viewpoint

Looking at the User viewpoint, two of its main goals are being considered for modeling: consult output cost details and produce reports. Seeing as these two goals only have in common the fact that they are perpetrated by the same viewpoint, they should merit two different goal models for their decomposition.

In what relates to the report production, it is a fairly simple goal to decompose: producing a specified report from the viewpoint of the user requires him to choose the report type he wants to produce and then input the parameters he is required to specify; these of course are specific enough actions to be considered requirements, however, according to the KAOS goal model notation, any requirement that is perpetrated by an agent outside of the system environment is considered an expectation, that is, these particular requirements are out of the system's control, and therefore the agents are expected to achieve them, not required. This decomposition is described in Figure 5-2, where the expectations are evidenced in a different graphical representation.

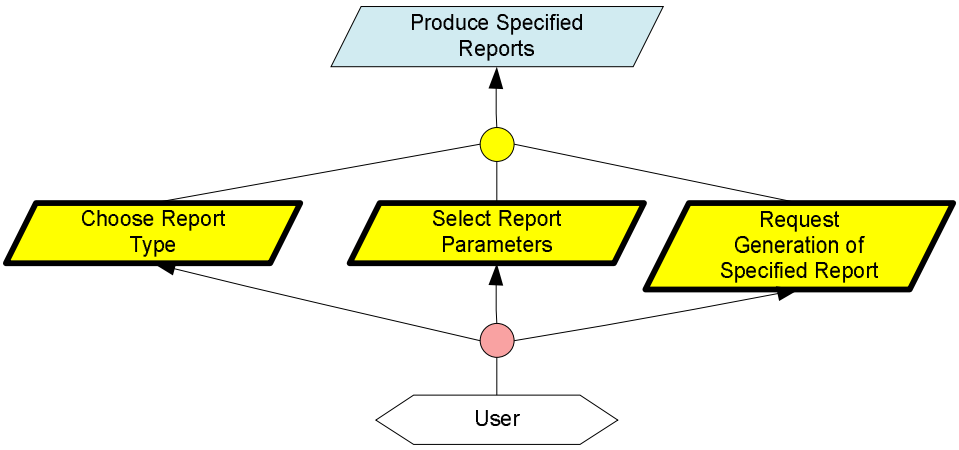

Figure 5-2 Produce Specified Reports Goal Model for the User Viewpoint

Regarding the User browsing of the cost details, this represents a more complex goal decomposition process. First of all, it is clear that the foremost goal is to output costs, that is, get them on display, which implies search parameters and the displaying of the results; search parameters in turn require both definition or selection and application, and although their definition is pretty straightforward, their application should be differentiated according to the output one requires, that is, whether they are to be applied to IPT/PBX, GSM or Asset costs, as is also differentiated in the project statement.

Thus, it is possible to conclude that the definition of filtering parameters constitutes a requirement for the goal of outputting costs, while applying them is a sub-goal that in turn is decomposed into three requirements for the three types of costs required; this can be seen in Figure 5-3. What can also be seen in Figure 5-3 is the fact that the outputting of results also

allows for the request of other formats to view the same results; this request of course should also be differentiated according to costs type and as such be reflected in the model.

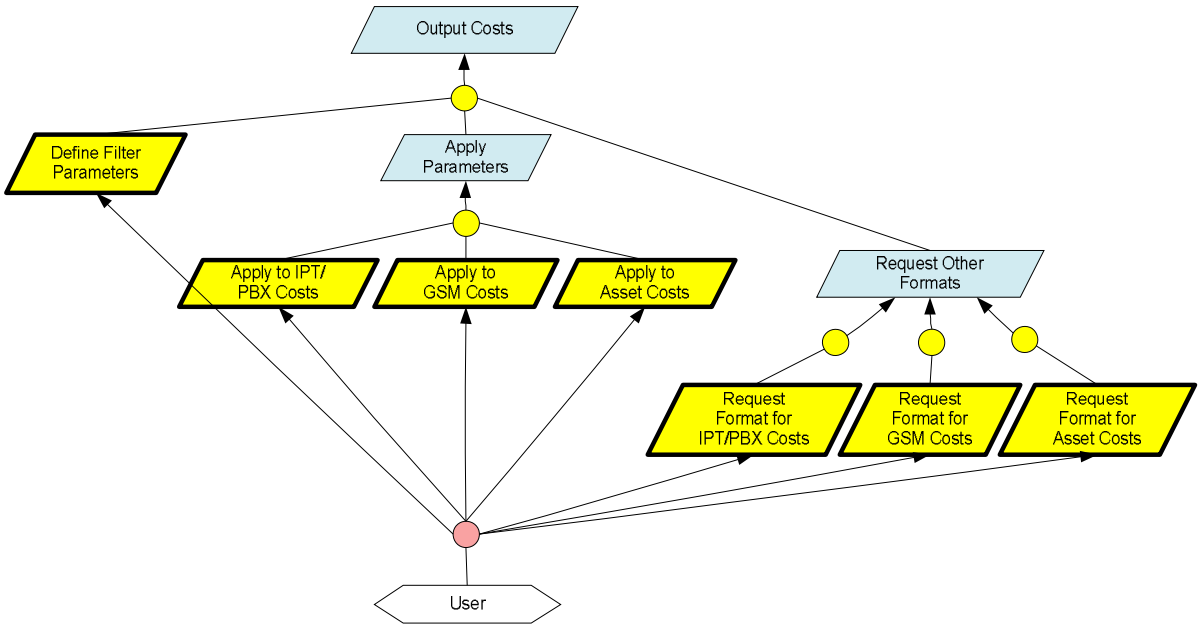

Figure 5-3 Output Costs Goal Model for the User Viewpoint

The remaining viewpoint is that of the Administrator. The goals chosen to be modeled are also diverse, as were that of the normal User, and therefore one should treat them separately.

The first goal to be considered is the management of the organization's hierarchical structure. This management is referred in the project statement as the addition, deletion or editing of the several organizational nodes that compose the said hierarchy; the first two can be immediately attributed to specific actions and thus expectations in the correspondent goal model. However, the editing of an organizational node entails several baser concepts as the simple editing of its details, changing its position in the hierarchical tree or managing the billable components associated with it.

Once more, the first two actions are specific enough to be granted expectation status, whereas the last can be further decomposed into the normal adding and removing operations, as well as an edition of the percentage attributed to each association between an organizational node and a billable component. These concepts and the above described decomposition are evidenced in Figure 5-4.

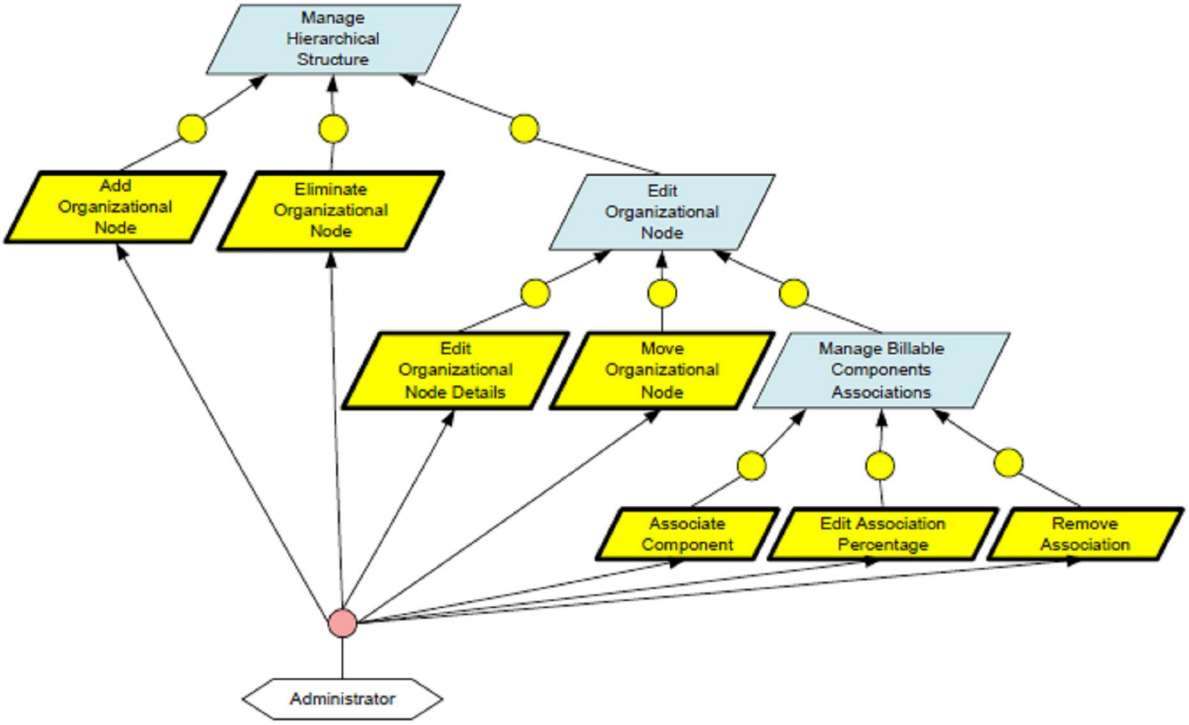

Figure 5-4 Manage Hierarchical Structure Goal Model for the Administrator Viewpoint

Once more, the UI interaction is patent in the model, seeing as the actions portrayed as expectations are UI-related. This nature continues apparent in the next goal to be considered: management of reports automation.

Although the concept of reports automation is that of an almost independent system that triggers scheduled emissions of emails to target users containing specified reports, there are still issues of configuration that are involved when dealing with such a system. As such, the top goal of managing reports automation can be decomposed into the creation and edition of those automations, as well as the activation of those already created. The activation process is straightforward enough to be considered a specific action, ergo an expectation, however both the creation and the editing of a report automation can be further decomposed and therefore should be considered sub-goals.

Figure 5-5 demonstrates how this decomposition is achieved: each of these sub-goals deals with three concepts that are referred in the project statement for a report automation, these being the definition of the report to be sent, the schedule the automation should follow and the

destinations that it should be sent to; these concepts are present both in the creation and in the edition stages.

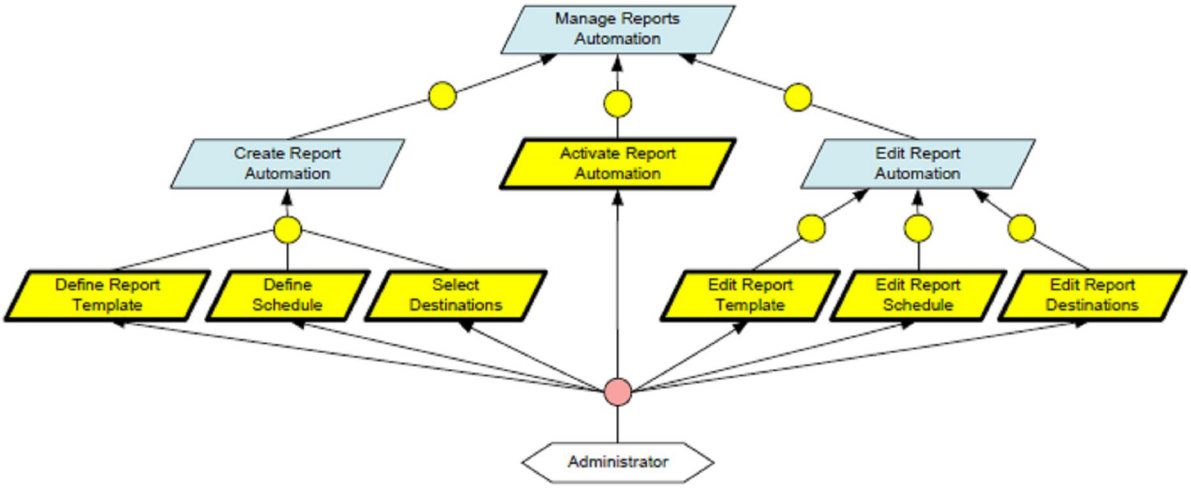

Figure 5-5 Manage Reports Automation Goal Model for the Administrator Viewpoint

The last goal to be addressed in this step is that of asset management. As one can see in Figure 5-6, asset management from the Administrator viewpoint is segmented: there are the entity-related actions that constitute base expectations for the administrator, and this same type of decomposition is apparent for the two sub-goals that correspond to both asset type and asset association management.

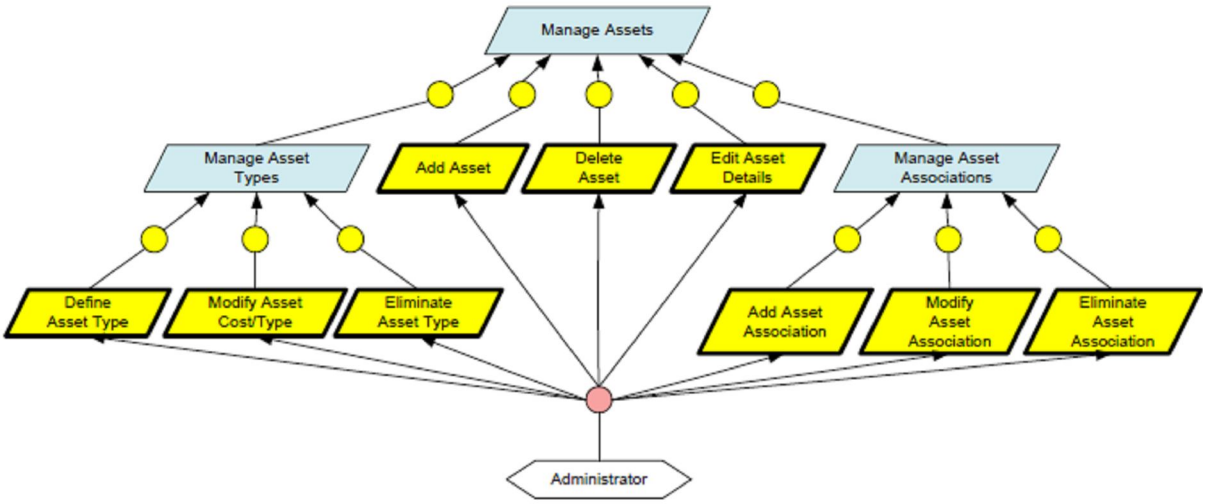

Figure 5-6 Manage Assets Goal Model for the Administrator Viewpoint

At this point the goal modeling step should be completed, taking into account that the remaining viewpoint goals are developed in a similar fashion, however there are still two concepts that although having already been referred, should be revisited.

5.4.1.2.3 Obtain set of Developed Viewpoints

After having developed each initial viewpoint's goal models, the analyst possesses a clearer notion of the system and the several agents and components that need to interact to guarantee the best configuration.

It is not the case with this particular case study, however it is important to remind that if one were to look at the list of system viewpoints realize that the list is too large to be handled confidently, it might be due to the a fact that in KAOS, agents are not limited to a number, which in turn is due to the fact that the Objectiver methodology deals only with models, and despite their complexity, models are usually easier to keep track of than textual representations, or particular entity development.

Furthermore, seeing as the hybrid approach is heavily based in PREview and focuses also on the AORE extension for comparative analysis, it is important that, as well as PREview did, this approach establish a limit for the number of viewpoints. Therefore, for complexity

management and maintenance of system perspective, the number of viewpoints would do well to keep to a 7±2 upper limit [16] and in order to formalize this transformation, they should be defined with more care as is explained in the heuristics chapter.

Before approaching the set of viewpoints to perform any changes one should first observe the set of stakeholder-related viewpoints. These naturally correspond to actual entities or interested parties in the system's development, entities that of course are not separable or even that do not belong in the same scope of interest. If one were to instance this train of thought to the case study, one realizes that indeed each of the two stakeholder-related viewpoints belong separated and should remain so.

Concentrating instead on the environmental viewpoints, one should once again take a look at them and identify those that make sense to group, or even divide, according to their sphere of action. Once more that is not the case with this case study, since the only environmental Viewpoint is that of the Telephonic Central, which is of course indivisible and there are no other viewpoints of the same type with whom to share a scope and merit a merger.

5.4.1.2.4 Elaborate each Viewpoint's set of Requirements

After these changes have been decided by the analyst, these new viewpoint definitions should be formalized into the tabular notation that has already been used with the initial ones. However, as was defined in the heuristic chapter, these finalized viewpoints should present a name, type, focus, sources of information, history and their requirements. In the case of the stakeholder-related viewpoints, the first two hold their initial definition, adding to it any changes that they might have suffered in the history section and collecting the identified requirements from their goal models (in this case requirements will correspond to expectations).

In the case of the environmental viewpoints, if there would have been any mergers or separations, their types would obviously stay the same, but their names should be defined during that transformation and a new focus should be defined either by joining or refining initial definitions. Furthermore, merger or separation history should be recorded in the new viewpoint's history while requirements should also be collected from goal models, but according to the new viewpoints scope.

In both cases, sources should be collected through the entire process and be detailed in the appropriate section of each viewpoint. Such was not the case with the sole environmental viewpoint: the telephonic central.

Regarding the collection of requirements from the goal models we can start by addressing the Telephonic Central viewpoint. Taking a look at figure A and applying the method explained for constructing the requirements textual description we would obtain the sole requirement definition:

"*The Telephonic Central is required to provide call detail records in order to produce database information.*"

Seeing as it is indeed the only requirement of this viewpoint, the finalized tabular representation can be constructed, outputting a result as seen in Table 5-4.

Table 5-4 Telephonic Central Viewpoint with Requirements

Name	Telephonic Central
Type	Environmental
Focus	Software or hardware components that handle telephony processing in the company and provide call information to the system.
Requirements	1. The Telephonic Central is required to provide call detail records in order to produce database information.
Sources	Telephonic Centrals re-sellers and manufacturers and respective documentation.
History	None

Observing the User viewpoint and its exhibited goal models, a larger set of requirements would be withdrawn and inserted into the tabular representation of the said viewpoint, as seen in Table 5-5. Regarding the "Produce Specified Reports" goal model, two requirements will be produced grouped under their parent goal, whereas in the "Output Costs" goal model, the case is more complex: there are two goal grouping levels, where the textual representation of those sub-goals should contextualize the case while making sense when read aloud. Therefore, according to the decomposition that sub-goal suffers, "the viewpoint is

expected/required to do action A and in order to do so…", followed by the requirements or expectations in order.

Table 5-5 User Viewpoint with Requirements

Name	User
Type	Stakeholder
Focus	The regular user of the application. Visualize output with minimal configuration possibilities.
Requirements	1. In order to produce specified reports... a. The User is expected to choose Report type. b. The User is expected to select report parameters. c. The User is expected to request report generation. 2. In order to output costs… a. The User is expected to define filter parameters. b. The User is expected to apply parameters, and in order to do so… i. The User is expected to apply to IPT/PBX Costs. ii. The User is expected to apply to GSM costs. iii. The User is expected to apply to Asset costs. c. The User is expected to request other formats and in order to do so… i. The user is expected to request format for IPT/PBX Costs. ii. The user is expected to request format for GSM costs. iii. The user is expected to request format for Asset costs.
Sources	Target test users of the application (regular company employees, department managers).
History	None

Regarding the Administrator, a table for the presented goal models would is present in Table 8-0-1 Appendix A displaying the same type of syntactic analysis of the corresponding models.

5.4.1.3 Produce and Develop the System Concerns

As was previously stated, although the concept of non-functional goal is present in many requirements engineering methodologies, in the PREview approach it has a special preponderance seeing as it drives the requirement gathering effort; in this hybrid approach however, the KAOS goal decomposition mechanism takes on that particular part.

The identification of non-functional goals, or concerns as they are called in the PREview approach, still represents a major part in the approach, seeing as they represent the inherent traits every system functionality should respect. However, contrary to what is stated in the PREview approach, it is my belief that, since non-functional requirements are many times difficult to identify, taking advantage of already developed goal models presents a useful stepping stone to obtaining a set of relevant concerns and their requirements.

By relying on the encapsulation mechanism viewpoints present, this identification, although still orthogonal in nature, can be successfully segmented into scopes of interest and still maintain a notion of the system in general by sharing common goals between those same viewpoints.

5.4.1.3.1 Produce List of Viewpoint Concerns

Producing a list of viewpoint concerns corresponds to the identification of the system's non-functional goals according to the scope of each viewpoint, i.e. inquire next to the sources associated with each viewpoint, registered previously in the tabular representation segment.

In what relates to the Br@in case study, although the registered sources were real, that is, despite the fact that during the project's development those sources were effectively considered and addressed to gather viewpoint information, it was passed on to this work's author by proxy through senior project analysts and they were the sources considered for this particular stage of the case study analysis.

According to the project's senior analysts and interviews and inquiries distributed throughout Br@ain's clients (using previous versions of the product), the foremost concerns for a system administrator (SA) are:

- Usability: a SA normally spends a great deal of time using the application and performing complex and tiresome tasks, therefore it is of great importance to maximize the productivity of the application and minimize the effect of repetitive actions, which implies usability. Furthermore, the system interface should be able to minimize also the complexity of the more arduous tasks.

- Response Time: once more, keeping in mind the total time that the SA spends interfacing with the application and the total number of actions that he is required to perform, a quick to respond system is essential.

- Availability: the vast majority of the SA operations are critical for the normal execution of the remaining system, thus their completion is mandatory; furthermore, many of those tasks are time-sensitive, like forcing certain scheduled executions, and so their availability should be assured.

- Security: as it has already been stated, a great deal of the administrator's tasks are system critical, it is safe to say that this status merits special permissions for accessing those particular system components; security is therefore an important issue, specifically relating to authenticating credentials.

- Correctness: issues of correctness are normally related with presented information, however when dealing with SA tasks, one recognizes that most of them involve mainly input; still, that input is required to be correct since the SA introduced information will serve as a basis for the system execution: entities details and associations with billable components, as well as scheduled tasks are critical systems regarding correctness of the data involved. Furthermore, seeing as an administrator contemplates all of the normal user's tasks, the description of this concern for said viewpoint also applies to this.

Regarding the environmental viewpoint known as Telephonic Central, its concerns derive from those expressed by the hardware manufacturers and technicians involved in setting up the environment for Br@in's interaction with these structures. Once more, these concerns were transmitted by proxy through the project's senior analysts:

- Compatibility: these hardware and/or software components are manufactured by different companies; also, they provide domain specific information that may vary between manufacturers. Therefore, although unification of collected data is a system goal, communication with the different telephonic centrals still requires specific connectors, in order to adapt to different developments.

- Security: when one is dealing with a delicate piece of hardware and/or software like a telephonic central, a great deal of factors can go wrong, which will produce an impact on the whole system and also the telephonic central itself. Therefore authentication is essential, as well as handling access permissions for each data connector the system will create for communication.

Finally, considering the User viewpoint, its main interaction with the application is that of data browsing: inputting parameters, obtaining search results, exporting the results; and reporting viewing. Thus the concerns that affect him are mostly UI related:

- Usability: as a focal point in the application's UI experience, the browsing manager expects a carefully studied approach to usability; although this viewpoint observes mainly search forms and tabular outputs, those forms are expected to be quite intuitive and direct the user along a logical path to obtain the required results; the results themselves are expected to be of easy consultation and indexation according to user needs.

- Response Time: although the quickness of response is not as important for this particular UI experience as it is for the administrator's, any UI experience should minimize tiredness and the wait period for a system's response to a user request, particularly the submission of search parameters.

- Availability: this may an important issue when dealing with companies whose regular users are mindful of their expenses and constantly check call details and costs. Although the system should be tailored for multi-access (as we'll see next), the system is expected to remain constantly available for any searches, no matter their volume; once more, the need for availability for a regular user is less than for an administrator.

- Correctness: since this viewpoint focuses mainly on the output, correctness is perhaps the most important concern to be considered, next to, obviously, usability; both call detail records and cost records should be carefully revised when processing data and if

necessary keep track of changes and processing logs in order to facilitate possible audits. Furthermore, reports are one of the main interactions between the user and the application; they are also the most commonly used basis for evaluation of particular departments (most department directors will not even user the browsing capabilities but instead will rely on the reports that may be automatically sent at specific periods); these reasons justify that correctness should be one of the highest concerns regarding this viewpoint, seeing as it should be assured at all stages of producing a specific report.

- Multi-Access: the scenario of a high number of users accessing the application's browsers is not likely; this tool is probably most appropriate for department managers than simple employees, and thus less likely to be used concurrently; however since heavy search procedures may be required, multi-access to the database to perform those searches should be assured.

- Security: issues of security regarding this viewpoint deal mainly with reporting and alerts privileges (ex: it could be a problem if regular employees could have access to particular administrative call records); special care should therefore be had during the setting up of destination lists for reports automations and for alert targets.

After consulting with the senior project analysts and taking into account the amount of concerns to be analyzed, it was decided that only four of them would be studied in depth, granted that these four would be considered the most important ones, according to the senior project analysts:

- Correctness, since it is the foremost concern of this application from the analysts' point of view.

- Response Time, due to the amount of tasks the administrator is required to perform as well as the department directors' (regular users) time spent using the application.

- Compatibility with the Telephonic Centrals is also of very high importance, seeing as a great deal of regular clients from past versions use Telephonic Centrals from different manufacturers and require the application to adapt to that fact seamlessly.

- Usability because there is a high probability that the client's users will not be used to this type of applications, as well as the designed administrators, therefore the learning curve must not be a steep one.

5.4.1.3.2 Develop Viewpoint Concerns Requirements

Having obtained a list of system concerns for the several system viewpoints, the hybrid approach's heuristics suggest the use of generic requirement patterns for handling the non-functional requirements that come from decomposing the discovered concerns. We will visit some of the patterns that were used for this particular case study and apply them to some of the identified viewpoints concerns, i.e. one per identified overall concern.

5.4.1.3.2.1 Verify Applicability of Generic Goal Patterns

Observing the Telephonic Central viewpoint and considering its concern, Compatibility, we are required to obtain a known compatibility pattern and apply it, instancing it to the activities developed in this viewpoint's scope, of if it is the case that such a pattern is not found or does not exist, provide one ourselves.

Compatibility is defined as the software's ability to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward-compatible with an older version of itself.

Seeing as the goal of this viewpoint is to provide call detail records and taking into account the project statement where it refers to the specific data connectors Br@in requires for each type of telephonic central as well as previous models of said types, one may try to contemplate that information on a requirement pattern (ex: Cisco Call Managers (the Cisco telephonic central) are constantly updating and various versions of that software are used in the several studied Br@in clients).

After careful research it was found that the compatibility issue has been addressed mostly in self contained code or modular components of self-contained systems. This however is not the case when dealing with the Telephonic Central. If we analyze the data we are provided in the

project statement and forwarded by the senior analysts, two generic compatibility goals emerge:

- It is a well-known practice for product manufacturers, be it a hardware or a software based product, to provide an API to allow other systems to easily interact with these components;
- If such is not the case, the interacting systems must be required to produce data connectors that adapt to the different products, their specific types and each type's specific model.

Transposing this to a generic goal model we have Figure 5-7. This pattern could be instanced to this case study regarding the Telephonic Centrals, but it could also be instanced to many other situations: interoperability with Microsoft Office tools, with Reporting Tools, etc.

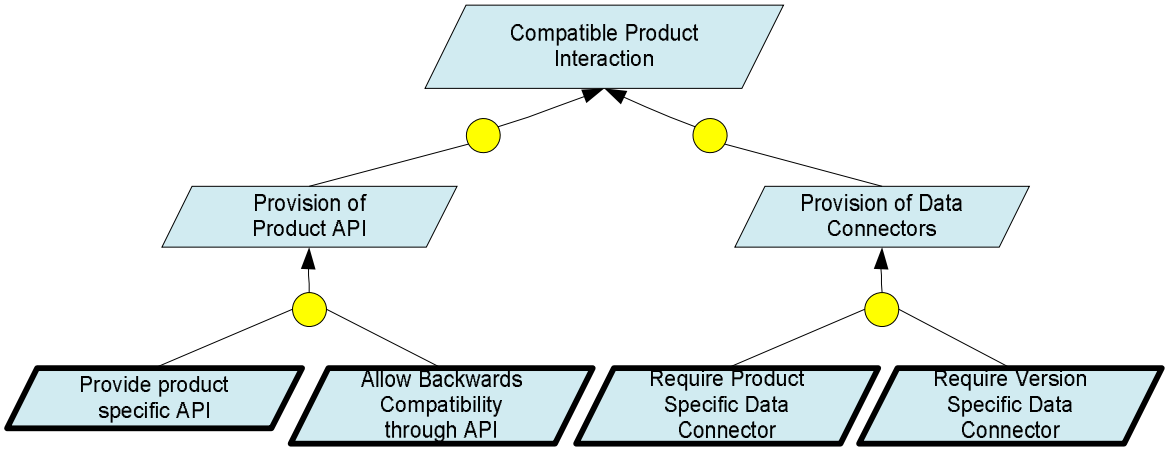

Figure 5-7 Compatibility Concern Generic Goal Pattern

As such, instancing this pattern to the Br@in case study, we obtain Figure 5-8, where we can see that if the Telephonic Central does not provide an API, as is the case with many manufacturers, the Br@in application is required to provide brand specific and version specific data connectors in order to enable a compatible interaction, i.e. uploading of call detail information.

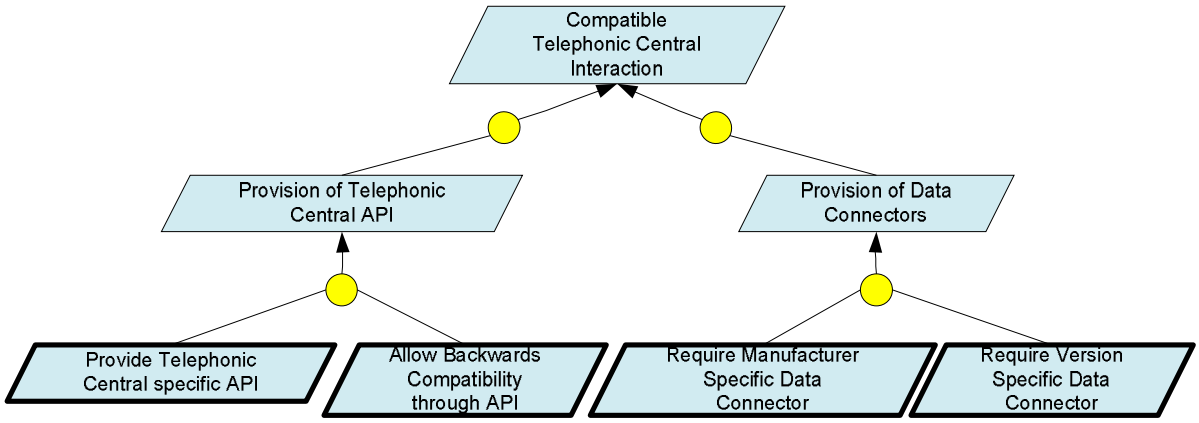

Figure 5-8 Compatibility Concern Goal Model for the Telephonic Central Viewpoint

Analyzing the User Viewpoint, we note that three of the identified concerns stand regarding the top four chosen for analysis. Focusing on the Correctness concern, the User expects the system's data browsers to output correct information regarding both call details as well as costs, and he expects correct report generation and exporting of browsed information.

Analyzing generic necessities of an application that produces "*browsable*" output and allows conversion of that same output to several formats, a generic goal model can be produced as seen in Figure 5-9. This model focuses on correctness as viewed by conformity with standards and conformity with minimal error margins of the industry.

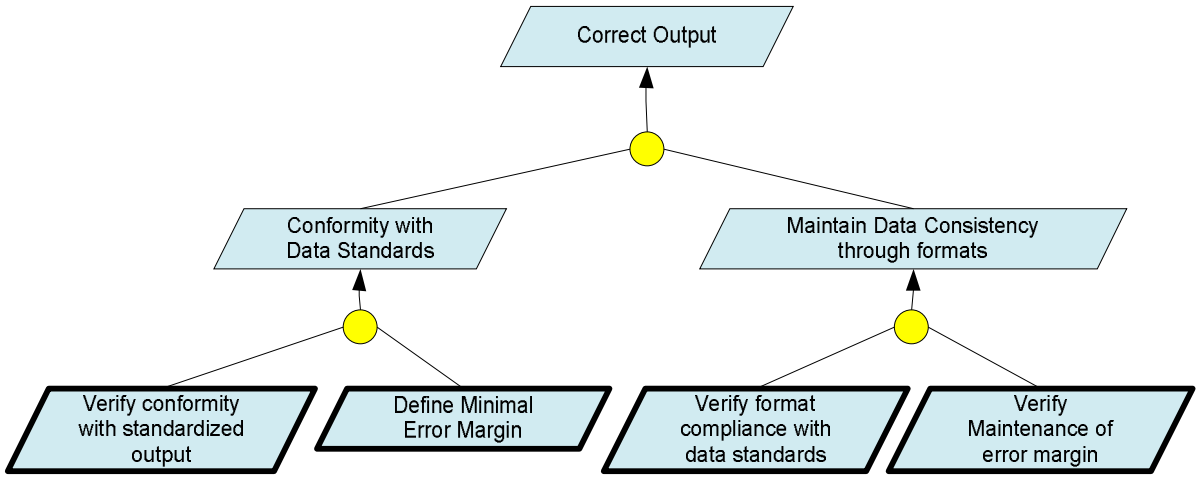

Figure 5-9 Correctness Concern Generic Goal Pattern

Instancing this pattern to the User viewpoint we have Figure 5-10 where both types of output are subject to appliance of this pattern, on the one hand focusing both on their conformity with Call Detail standards as well as with spreadsheet standards, while maintaining a 0% error margin policy, and on the other hand focusing on conformity with report templates and reporting tools minimum requirements.

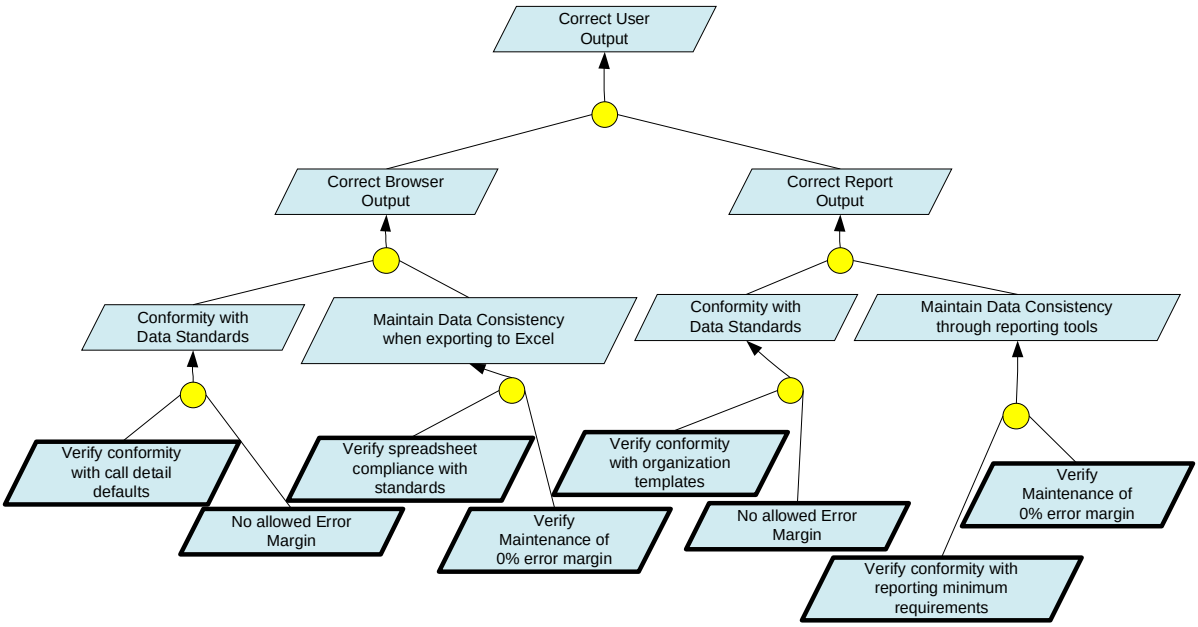

Figure 5-10 Correctness Concern Goal Model for the User Viewpoint

Observing the issue of Usability, there is a pattern described in [21] that is emulated in Figure 5-11. This particular pattern does not extend its analysis to requirement leafs but instead allows its instances to define how to do so.

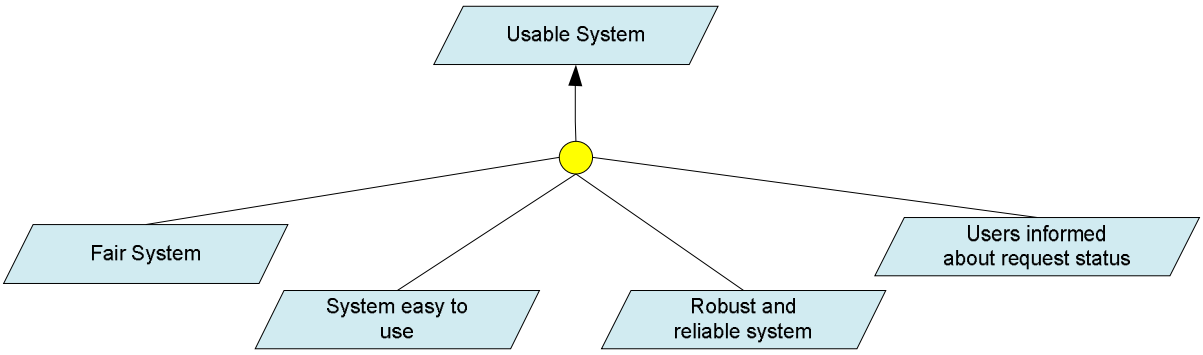

Figure 5-11 Usability Concern Generic Goal Pattern

On the other hand, analyzing other documentation regarding usability patterns [10] we can further decompose this pattern into a more concrete one. According to [10], one should decompose usability traits or attributes into usability properties that provide help regarding those attributes and furthermore identify patterns that derive from said properties.

Focusing on figure K sub-goals, we can ask “what is a fair system?” where we could say that it is one that is consistent and grants status-equivalent access time; transposing this analysis to the “easy to use” attribute, we can state that in order to do that, a system must guarantee guidance for specific tasks and natural mapping of those tasks in the interface; a robust and reliable system is one that focuses on error prevention and access policy control and at the same time maintains a functional and evolutionary consistency; finally, keeping users informed means providing feedback on their tasks and warning about erroneous situations.

This analysis instanced to the Administrator Viewpoint, and considering its goals in a generic light, translates into figure L.

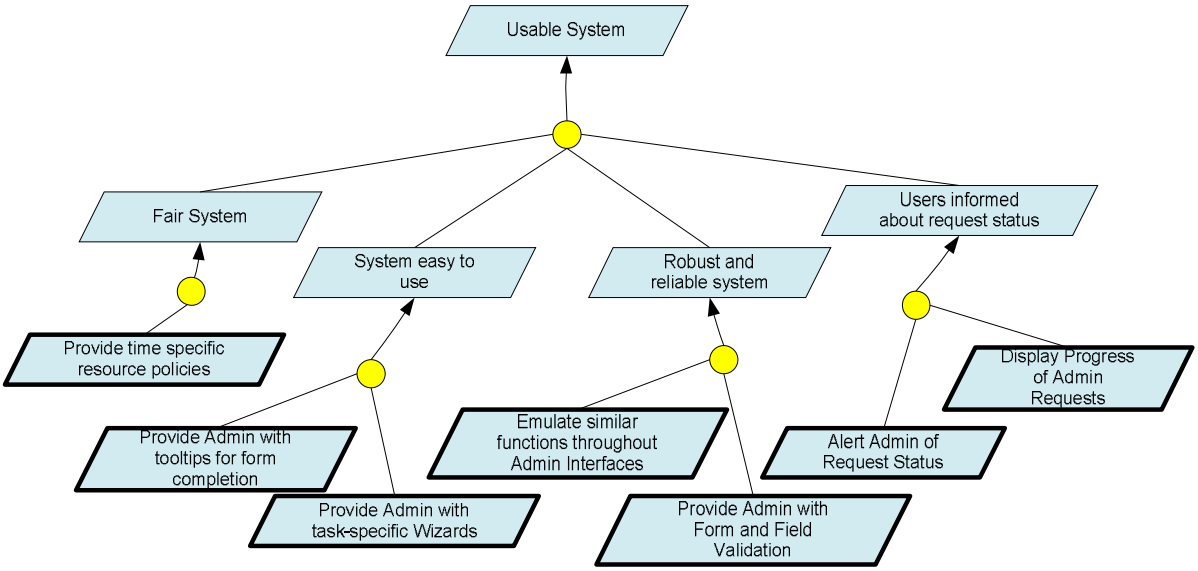

Figure 5-12 Usability Concern Goal Model for the Administrator Viewpoint

Finally, considering the Response Time concern, we can consider the example of the W AJAX design pattern [4] where the interface is rendered while the results are still being obtained, as well as adding some input regarding database access and browser common knowledge, obtaining Figure 5-13.

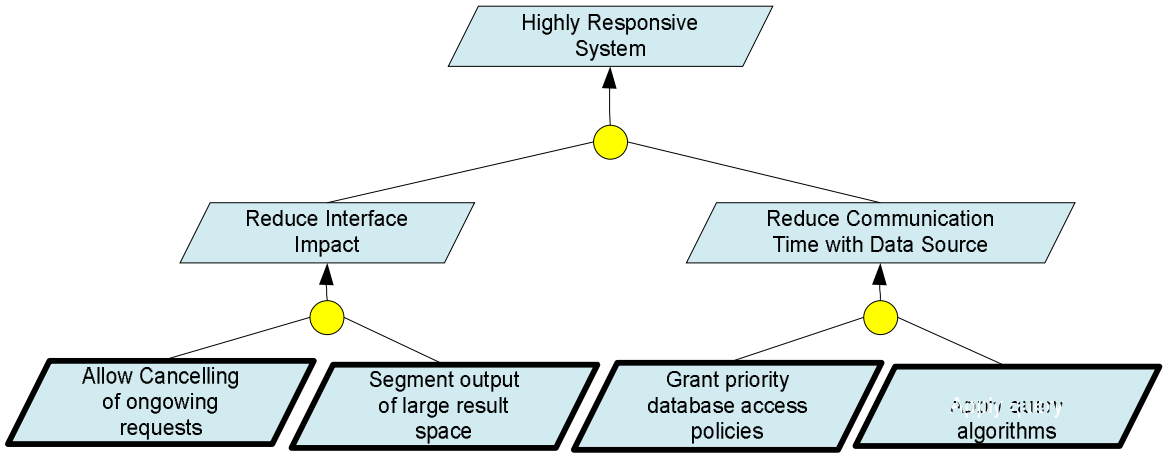

Figure 5-13 Response Time Concern Generic Goal Pattern

This pattern applied to the User viewpoint is particularly interesting since most user requests pertain to Department Directors and the amount of records to be consulted by a Department

Director for example is sometimes immense. Segmenting those results into smaller sets that could be uploaded to the UI in short bursts would allow the user to start consulting them while the rest are being loaded. Also, granting temporary “all-access passes” to department directors during their searches allows optimizing their queries with minimal impact to concurrent regular users, whose result space is normally smaller and thus less resource consuming. This is seen in Figure 5-14.

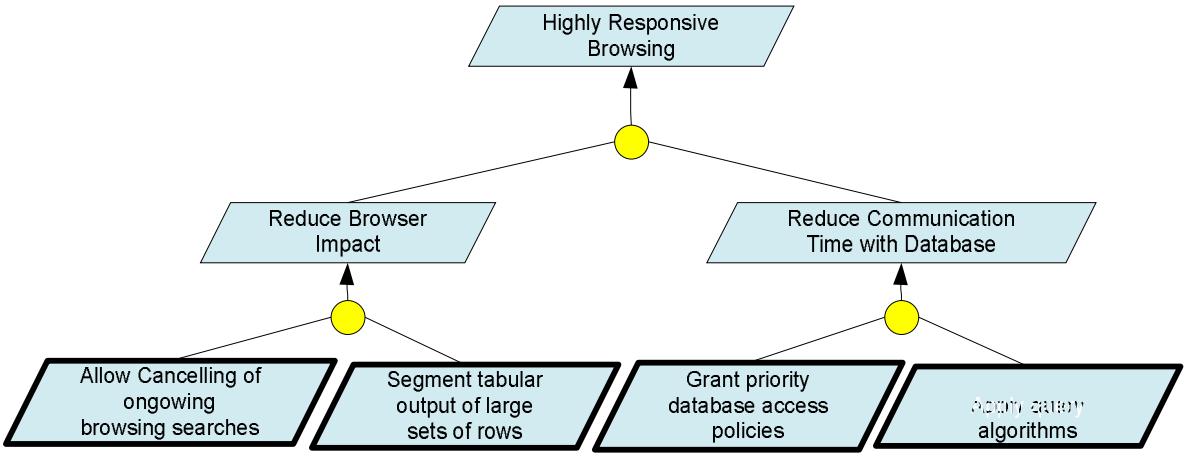

Figure 5-14 Response Time Concern Goal Model for the User Viewpoint

5.4.1.3.2.2 Merge System-Wide Concern Goal Models

Finalized the concern analysis for each viewpoint there is still the fact that Concerns are orthogonal in nature, and as such need to be contemplated from a system-wide point of view. Taking advantage then of the softgoal models developed for each viewpoint, and relying on the fact that viewpoint segmentation allows for a full scope of the system when totalized, it is safe to assume that merging the several viewpoint specific concern models according to each concern, will produce an accurate system-wide view of that concern.

The compatibility concern is presented only in the Telephonic Central viewpoint and therefore its view is already system wide as portrayed in Figure 5-8. However other system concerns, as is the case of the Response Time concern, can only be viewed in system-wide perspective when considered from all of their viewpoints. A merged concern model can be seen in Figure 5-15 where both the administrator and the user viewpoints provide input on

this concern. The user viewpoint's input more reliant on reducing the time spent in large result space queries, and the administrator focusing on simple UI maximization by working with asynchronous data sources.

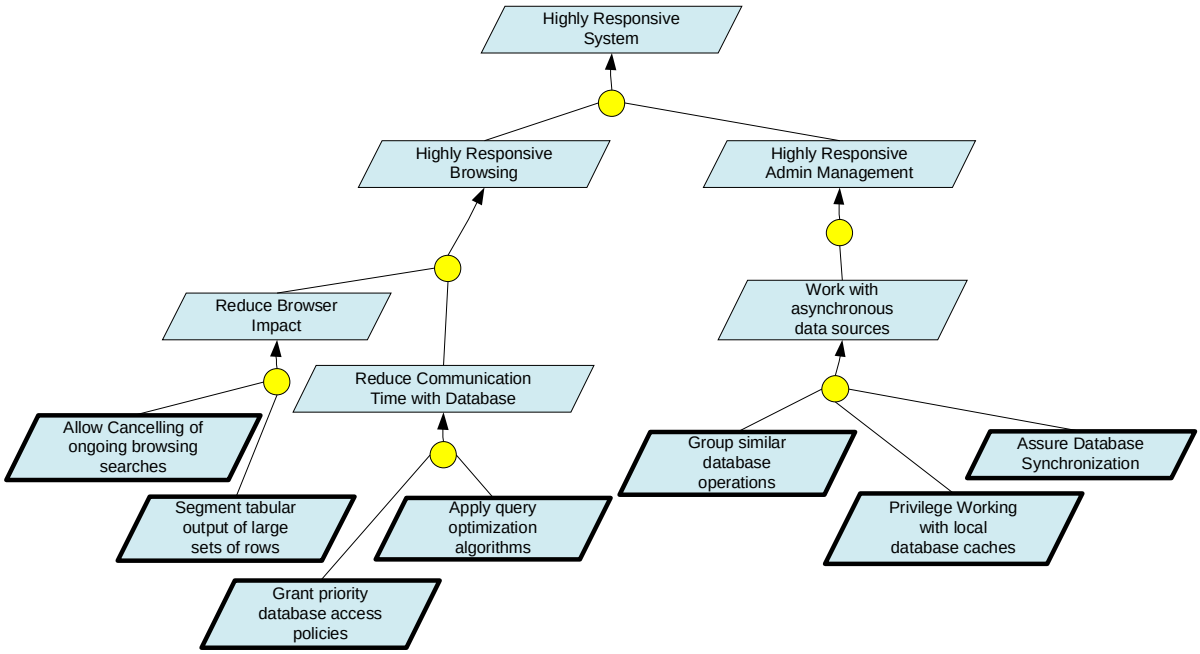

Figure 5-15 Response Time Concern System-Wide Goal Model

5.4.1.3.2.3 Register Concern Requirements

Regarding the model drawn for the Response Time concern, its requirements would be registered in Table 5-6, where the same textual quasi-algorithm is followed to produce the requirements to be inserted.

Table 5-6 Response Time Concern Tabular Representation

Concern	Response Time
Affected Viewpoints	Administrator, User
Requirements	1. In order to achieve a highly responsive system… a. The system should achieve highly responsive browsing and in order to do so… i. The system should reduce browser impact and in order to do so... 1. The system should allow cancelling of ongoing browsing searches. 2. The system should segment tabular output of large sets of rows. ii. The system should reduce communication time with database and in order to do so... 1. The system should grant priority database access policies. 2. The system should apply query optimization algorithms. b. The system should achieve highly responsive administrative management and in order to do so… …

References to those requirements should then be added to the viewpoints they pertain to, as is shown in Table 5-7.

Table 5-7 User Viewpoint Tabular Representation with Concern Requirements

Name	User
Type	Stakeholder
Focus	The regular user of the application. Visualize output with minimal configuration possibilities.
Requirements	...
Concern Requirements	1.a.*
Sources	Target test users of the application (regular company employees, department managers).
History	None

5.4.1.4 Represent the System's Obstacles and their Solutions

Concerning this step in the Hybrid Approach there is a particular concept that should be referred: viewpoints concern entities that are foreign to the system, they interact with the system and regarding the goal models that are designed to obtain their requirements, their interaction with the system implies that they are expected to perform particular actions (in the case of stakeholder viewpoints) or are required to perform particular actions (in the case of environmental viewpoints). This does not extend however to how the system can and should mitigate certain aspects of its normal functioning, therefore this obstacle analysis will limit itself to obstacles the viewpoints themselves will encounter when interacting with the system and how they should respond to those obstacles.

Applying this step to the Br@in case study, more precisely to the User viewpoint and its browsing of call details, we verify that the only error prone point would be the generation of output on behalf of the system. Barring any malfunctions, the only error would be the lack of search results, either due to search parameters that were too strict or by loss of connection to the database. On both accounts there is a possible response from the User: to the first he can re-define search parameters, to the second he can either wait a few seconds and try again or report this lack of connection to the tech support. This is shown in Figure 5-16.

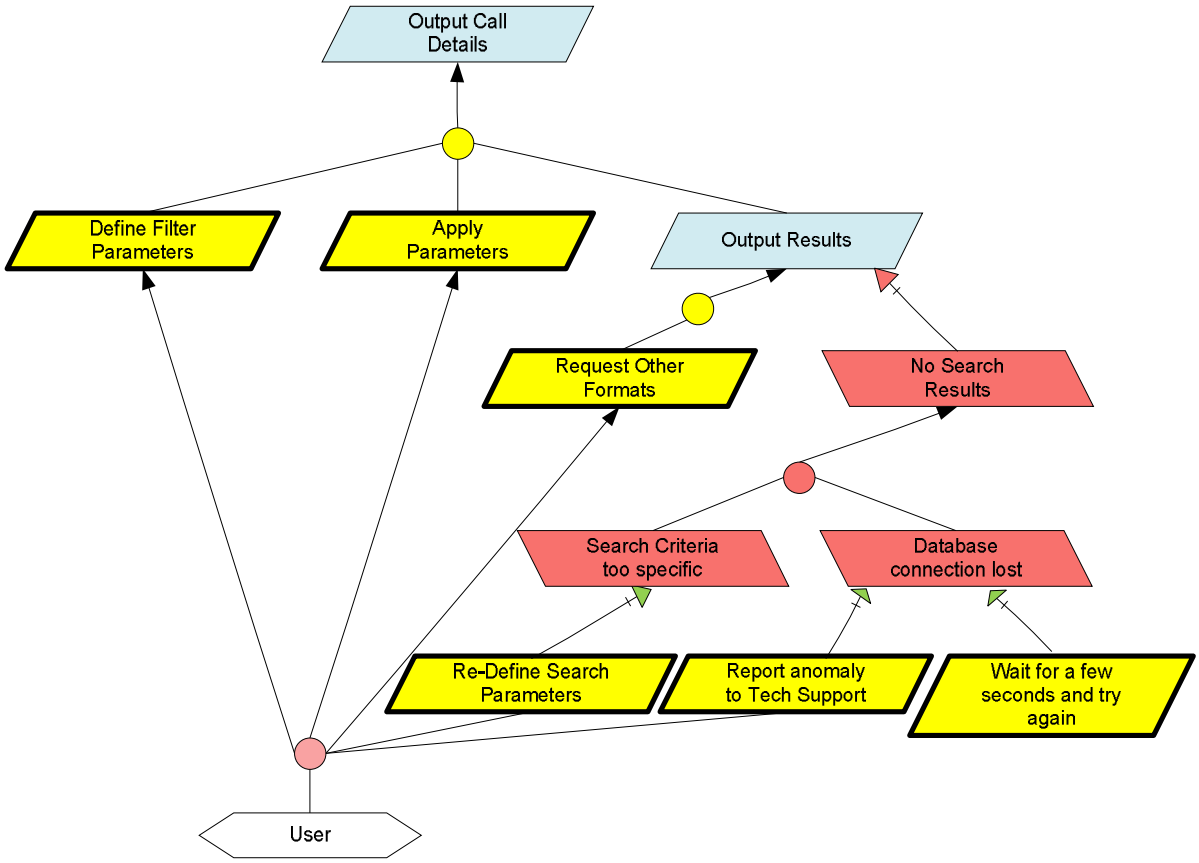

Figure 5-16 Output Call Details Goal Model for the User Viewpoint with Obstacles

Throughout user interaction with the system these are the obstacles he should be faced with: either while browsing cost details or even when generating reports. Regarding the reporting tools, the user expectations are very clear as seen in Figure 5-2: reports are selected by type, then their parameters are also selected, not defined, and therefore this leaves little room for user-derived errors; thus the only error-prone point would also be the generation of said reports regarding their contents, and to which the user responses would be similar to those referred above.

Regarding the Administrator and its managing of database entities there is a specific error prone point pertaining to licensing limits. The Br@in application will obviously impose licensing limits regarding its interaction with the database, that is, according to senior analysts consulted for the purpose, a client would purchase a Br@in license and according to the contract, that license would limit the number of entities of each type to be considered.

Looking at the goal model regarding the Administrator's management of the system operators, we verify that when adding an operator to the system, there is a likelihood that the

license limit may be infringed, thus an obstacle to this requirement would be said infringement. Another obstacle might appear related to dependencies between database entities: although call details and cost details that make up the information Br@in processes may refer to an operator, information which is not bound to that operator's existence. However, there are tariff plans and other database relevant information that may depend on the operator's existence; thus when eliminating it, some dependencies might constrain that action. These obstacles and their solutions are demonstrated in Figure 5-17.

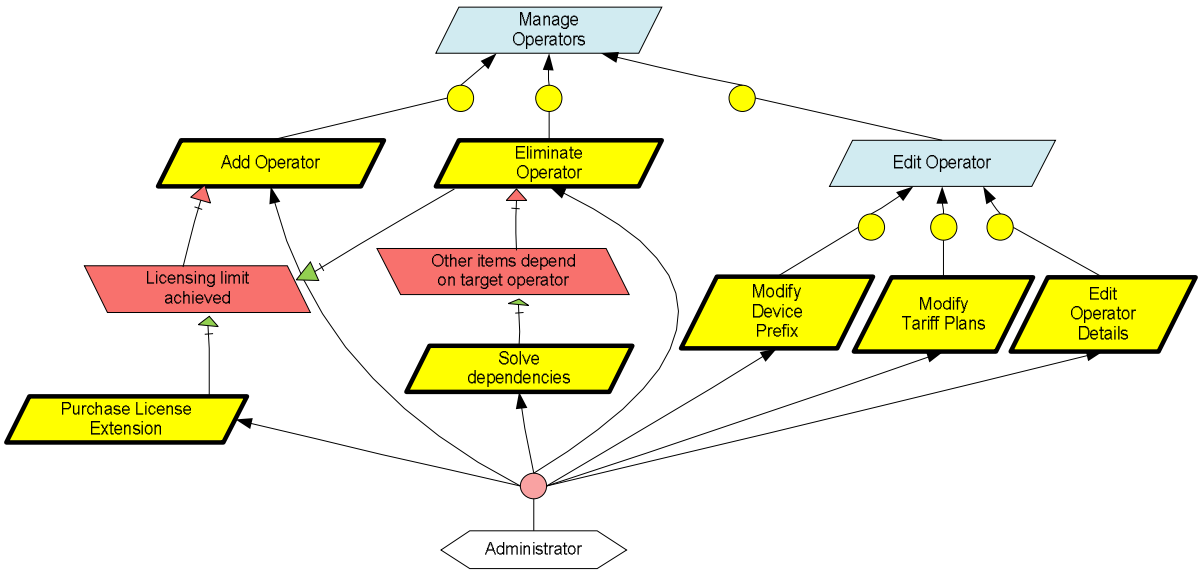

Figure 5-17 Manage Operators Goal Model for the Administrator Viewpoint with Obstacles

We now a look at the Telephonic Central in Figure 5-18 to complete a viewpoint-wide analysis of possible obstacles that might impede system functions and how the viewpoints regard those obstacles and their solution. While performing the only interaction with the system that the Telephonic Central perpetrates, that same interaction known as collection process, might be terminated abruptly, which can happen for two reasons: first and most obviously due to a communication failure, in which case the Telephonic Central is required to cache the records that it was sending and wait for a re-connection attempt; on the other hand, during the collection process, a licensing limit might be reached concerning the call details that the application is allowed to import to the system, in which case the telephonic central is required to warn system of licensing limits and advice license extension.

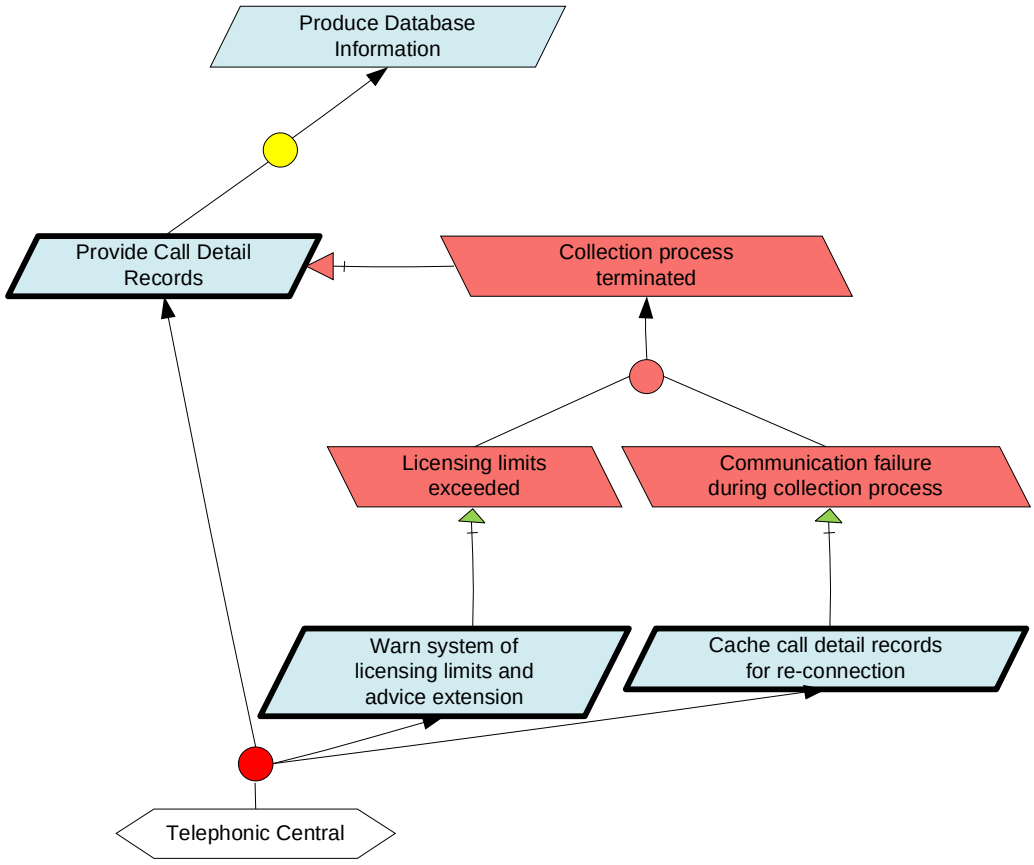

Figure 5-18 Produce Database Information Goal Model for the Telephonic Central Viewpoint with Obstacles

Finalized this stage of modeling the obstacles for the several goal models each viewpoint presents, it is required that those obstacles and their solutions be registered in the tabular representation of each viewpoint for documentation, as is shown in Table 5-8 for the Telephonic Central viewpoint. As we can see, when faced with composite obstacles, that is, obstacles that are decompositions of more generic ones, a full context textual description is required by traversing the obstacle nodes.

Table 5-8 Telephonic Central Viewpoint Tabular Representation with Obstacles

Name	Telephonic Central	
Type	Environmental	
Focus	Software or hardware components that handle telephony processing in the company and provide call information to the system.	
Requirements	1. The Telephonic Central is required to provide call detail records in order to produce database information.	
Concern Requirements	…	
Obstacles	Opposed Req.	1.
	Obstacle	Collection process terminated due to licensing limits exceeded.
	Solution	S1.1. The Telephonic Central is required to warn system of licensing limits and advice extension.
	Opposed Req.	1.
	Obstacle	Collection process terminated due to communication failure during collection process.
	Solution	S2.1. The Telephonic Central is required to cache call detail records for re-connection.
Sources	Telephonic Centrals re-sellers and manufacturers and respective documentation.	
History	None	

5.4.1.5 Requirement Analysis

Requirement analysis is a particularly important stage of the PREview approach. It is a three step comparative take on the elicited viewpoints, concerns and their requirements and has the intent of clarifying the elicited components by filtering those that are erroneous or overlapping, while clarifying the importance of those that are quantifiable.

5.4.1.5.1 Perform Inter-Viewpoint Interaction Analysis

Inter-Viewpoint analysis is a step of the Hybrid Approach that although conceptually interesting is maintained from the PREview approach more as a precaution than for any other reason. Viewpoint requirements are functional tasks the system viewpoints are required or expected to perform; their functional nature does not allow establishing priorities seeing as they either are performed or they are not.

As was said while describing this approach's heuristics, this is a step that is particularly important if one considers that, in a team effort, each viewpoint's requirements might be elicited by different team members. As such, it would be of extreme importance to compare the different requirements and verify if they indeed conflict or overlap and re-design is required.

Seeing that this case study was developed by this thesis' author, there was throughout the process a persistent knowledge of each viewpoint's requirements and how they should not conflict with other viewpoints that were elicited. Due to that fact, realizing this analysis would likely output no relevant results. However, in the interest of demonstrating the approach's applicability to industry case studies, we present Table 5-9.

Table 5-9 Inter-Viewpoint Comparative Analysis (Admin. vs. Tel. Central)

		Administrator								
		1.a	1.b	1.c.i	1.c.ii	1.c.iii.1		X.a	X.b	X.c
Telephonic Central	1	-	-	-	-	-	…	-	-	-
	S1.1	-	-	-	-	-	…	-	-	-
	S2.1	-	-	-	-	-		-	-	-

In Table 5-9, an inter-viewpoint analysis is conducted between the administrator and the telephonic central. These two viewpoints have a possible conflict point when it comes to managing the collection process on behalf of the administrator and executing said process on behalf of the telephonic central. The first requirements seen in the table from the administrator's input are regarding the management of the hierarchical structure, it would

follow on detailing every requirement's reference until it reached a hypothetical X requirement branch that would represent the three expectations of the management of data processes from the administrator viewpoint. Although these requirements would have been the most probable source for errors regarding this analysis, such does not happen.

5.4.1.5.2 Perform Inter-Concern Interaction Analysis

Inter-Concern analysis is yet another entry point for KAOS contributions, this time merged with a contribution from the AORE approach [20]. The overall concern model that is suggested in this approach's heuristics represents the conflict and cooperation that might exist between the several system concerns, analyzed at a viewpoint specific depth.

For the Br@in case study, such a model is seen in Figure 5-19, where the concern contributions are clearly identifiable by the "+" and "-" symbols, detailing their conflict or cooperation. This particular set of concerns presents a great number of positive contributions between concerns which is a very good sign: it means that the stakeholder needs have an overall positive contribution to the system's development and few concessions will have to be made in order to obtain a stable and final set of system requirements.

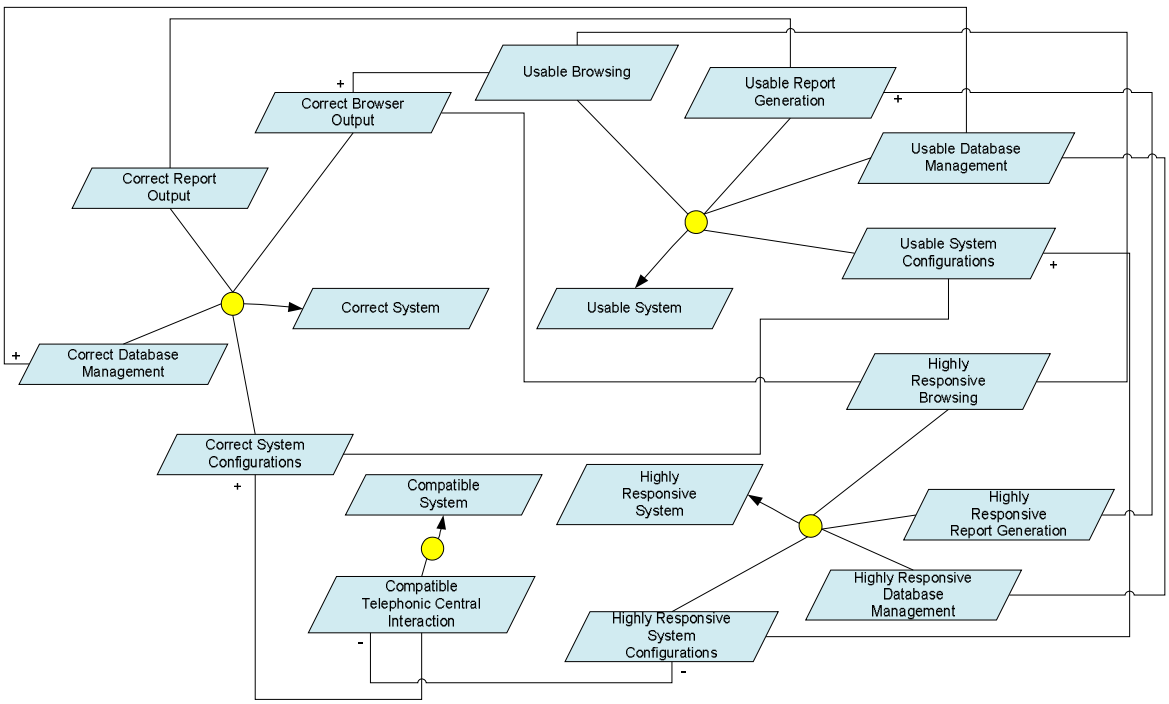

Figure 5-19 Inter-Concern Interaction Analysis

5.4.1.5.3 Perform Inter-Concern Interaction Analysis

Although this step may be considered a summarizing one, it constitutes an important stepping stone for the final stage of the approach. We are required to produce a Viewpoint-Concern interaction matrix that summarizes the relationships between each viewpoint and the several system concerns, in order to provide a basis for the previously referred final stage.

Regarding the case study, if we consider the different viewpoints and concerns considered and analyze their respective tabular representations, we obtain Table 5-10. There we can see for example, that this limited scope of concerns outputs similar characteristics for both the User and Administrator viewpoints.

Table 5-10 Viewpoint-Concern Interaction Analysis

	User	Administrator	Telephonic Central
Compatibility			X
Response Time	X	X	
Usability	X	X	
Correctness	X	X	

5.4.1.6 Requirement Negotiation

Entering the requirements negotiation stage of the approach means that most of the analyst's work has already been done and all that remains is to analyze inter-concern relationships and their co-existence within each viewpoint's scope, and negotiate compliances for potential conflicts.

5.4.1.6.1 Produce a Weighted Contributions Table

As is presented in the heuristics section, this final stage of the approach consists of consolidating an analysis that has been performed in previous steps, along with gathering stakeholder input in order to resolve conflict that might emerge.

The limited scope of the considered concerns and the nature of the concerns themselves will of course limit this analysis; however there is still a sufficient sample to present relevant results. The large amount of positive contributions present in Figure 5-19 is not reflected in this last analysis, which is due to the fact that the goal is to solve conflicts and of course cooperation does not require the establishing of priorities. If we focus on the conflicts and verify those that exist in the same viewpoint, we verify that in the case of correctness and response time, where correctness efforts can have a detrimental effect on the system's response time, there is a conflict, and seeing that is occurs in the scope of a viewpoint it must be settled.

This weight attribution is obtained by presenting this conflict to the stakeholders for them to resolve it by defining priorities. This analysis is present in Table 5-11.

Table 5-11 Weighted Contributions Analysis

	User	Administrator	Telephonic Central
Compatibility			X
Response Time	0.8	0.8	
Usability	X	X	
Correctness	1.0	1.0	

5.5 Summary

The goal of this chapter was to demonstrate this Hybrid Approach's capabilities, as they were described in the previous chapter, in a real world case study. The intent was to carefully follow the identified steps and in the end obtain a stable set of functional requirements organized according to viewpoints, and non-functional requirements organized according to concerns.

Furthermore, there was also the intention of demonstrating that regarding the PREview [23] approach, the additional components this hybrid approach included would result in a clear gain in what concerned requirements discovery both from the functional and non-functional sides of the system. Requirement analysis and negotiation were also intended to be better addressed as a result of both AORE [20] and KAOS [13] contributions.

From the onset of applying the approach to this case study it was clear that it would present a different challenge than that of the Via Verde case study: firstly because it was a real industrial application, and second because it was a different type of system.

The Via Verde case study produced a large set of viewpoints, even in a finalized state, this identified it as somewhat of a distributed system, that is, a system composed of many foreign components, and therefore, perspectives. This however was not the case with the Br@in case study.

Br@in is a self-contained application that interacts mostly with two types of users and a possible set of telephonic centrals that provide the call data for the application to process. This implies that most of the work is done by the system itself and thus cannot be modeled using an approach like PREview. Another fact to consider is that although having fewer viewpoints, each of this system's interacting agents was responsible, for the most part, for a large set of tasks, which represented another change regarding the Via Verde case study.

Iterating through the approach's steps revealed itself to be intuitive and always produced a result set, which of course reflects the analyst's progress in a quantifiable manner. Providing clear heuristics for obtaining the system's requirements, both functional and non-functional, produced a precise set of requirements that required little rework and avoided some of the inevitable iterations that a subjective approach like PREview would have implicated.

Reaching the last stages of the approach, the analysis of the obtained requirements was achieved also in an intuitive manner, with a minimum of overlapping or conflicting requirements between viewpoints and a clear exposure of conflicts and contributions between system concerns. This in turn provided the analyst with a last useful tool with which to discuss with the project's stakeholders (in this case, senior project analysts from the Br@in project) regarding the establishment of priorities for each concern pertaining to each viewpoint's scope.

Chapter 6
Conclusions

I have worked with both the PREview and the KAOS approaches in academic contexts for several times; in all accounts, both approaches seemed well cut out for the job they were devised to develop and although very different in nature, both adjusted to the work and the projects they were used for. The nature of this work was born precisely from the usage of the two approaches and the issues identified with using each of them, mostly the PREview approach.

This work began by presenting a brief analysis on two Requirements Elicitation and Analysis inclinations: viewpoint oriented and goal oriented. Several approaches were presented on either account, with special emphasis on one from each inclination: KAOS and PREview. Each approach was then described in detail, focusing on their main traits and the complementarities of both approaches with an emphasis on directing the analysis for an attempt at merging these two techniques, analyzing the benefits of such an attempt.

Considering the issues identified in each approach, it was verifiable that both approaches were complementary. On the one hand, KAOS offers a set of requirements elicitation heuristics through goal decomposition, however lacks a means to organize the elicited requirements in a stakeholder-friendly fashion. On the other hand, PREview focuses on the understanding and controlling the complexity of these systems by separating the interests of various stakeholders and formalizing this multi-perspective view into analysis methods, however lacks a mechanism to guide the elicitation process and thus relies heavily on the analyst's abilities and forcefully, his experience.

The objective of this thesis was therefore to propose a hybrid approach that brought together the advantages of both of the base approaches, however focusing on the PREview approach as a basis, since it was the one which had more to gain by incorporating components from the other approach. The result would be synergetic where, for example, completion is better addressed by providing heuristics to the process of requirements elicitation.

6.1 Contributions

On a first look, it might seem that this Hybrid Approach adds aspects and routines to the processes of an already effective approach. However, if one is to look at both of the base approaches, it is clear that PREview is a much more subjective approach, as KAOS is a more systematic approach, or too formal. This Hybrid Approach is therefore an attempt at finding a half-way point between paradigms, taking advantage of each approach's benefits and attempting to focus on mitigating PREview's lacking.

As it is shown during the course of this work and mainly through the case study analysis, the inclusion of the KAOS heuristics into the PREview process of identifying viewpoint requirements greatly simplifies the work of the analyst, as it is simpler to follow clearly defined steps when comparing to subjectivity. Furthermore, the complex nature of the KAOS approach does not appear as an intruder, seeing as it is greatly reduced when approaching system design considering each viewpoint in turn. It might add some redundancy to the existing processes, seeing as several viewpoints might share similar requirements, but it will treat larger knowledge bases in a far more informed manner.

The inclusion of the AORE approach when dealing with conflicting requirements and concerns is an important contribution to the approach as it provides an informed input into the negotiation process. Its translation into KAOS goal models with extended notation to reflect concern conflict and cooperation proves to be a good source of information when producing informed documentation for negotiation with project stakeholders, both relying on KAOS' graphical capabilities and the AORE inspired extended notation's compliant input.

It is fair to say that, although this approach implies a significant overhead regarding documentation and modeling, the advantages that come from taking such a subjective approach like PREview and extending it with a set of requirements discovery heuristics, a set of methods for requirements analysis and a basis for requirements negotiation, far outweighs the effort it implies. Furthermore, despite KAOS' formal nature, its graphical notation in the form of the Objectiver Methodology represents a lightweight manner to include the same

principles of formal mechanisms the approach stands for. This maintenance of a lightweight nature was of the utmost importance, seeing as it was one of PREview's main goals.

As was said before, this thesis proposes a hybrid approach and describes its possible application to both a simple but descriptive case study, when explaining its heuristics, and a more complex and real world industry case study, developed in a corporate environment, and thus demonstrative of the industrial applications of this Hybrid Approach for detailed analysis.

Its successful application to the real world case study reveals it not only as a theoretical approach, but also as one that might have interesting usage in an industrial environment.

6.2 Future work

Possible evolutions for this approach consider firstly technical aspects that could be experimented in order to deem advantageous.

An aspect that was not considered while detailing viewpoint requirements decomposition was that of, while interpreting the KAOS goal models, considering the AND and OR branching differences and contemplating them in each viewpoint's tabular representations. Although their importance is notable in a graphical model, benefits regarding the listing of a viewpoint's requirements with such detail are not clear.

Viewpoints relate to entities that interact with the system, directly or indirectly, but that are not part of the system; the KAOS approach however extends this analysis to also contemplate system components as agents. Extending the notion of viewpoint to envelop system components as modules could perhaps contribute with additional knowledge of the system, with the counterpart of adding more overhead to the original PREview approach.

Applying this hypothesis to the Br@in case study, we could obtain modules responsive for handling the UI back-end of each user's interaction with the system, demonstrating how to handle error-prone situations, not only from the foreign viewpoints' side, but also from that of the system itself. These Agents would not be viewpoints per se, seeing as a viewpoint's definition is exactly that of a foreign agent, but the notion of an encapsulated development sub-space could produce interesting results.

It is an obvious evolution for any development technique that it should develop tool support for the theoretical knowledge it provides; even more so if it is a modeling technique with graphical components.

The KAOS approach is already supported by the Objectiver tool for goal modeling and the subsequent models that the KAOS approach entails, however for the PREview approach, to current date, there is no official tool support, which makes sense since it is a heavily subjective and textual approach.

For this work though, the Objectiver tool was not used, but instead the Microsoft Office Visio 2007 tool. For the specific purpose of developing goal models, a new stencil was created at the early stages and used throughout the development of both case studies. Any tool support for this approach would therefore require a hybrid tool for both modeling and textual support, that is, a modeling tool similar to Visio or Objectiver for the goal models and a form based tool for Viewpoints documentation. Introduction of an XML notation instead of a tabular representation for registering Viewpoint and Concern information could demonstrate usefulness when registering form information in a data source.

Another interesting development would be integration with previous work developed by myself and colleague Ana Sofia Penim for the Semantic Web course in 2008 [19], concerning Ontology support for the KAOS approach: similar ontological support would be a useful complement for this approach since it relies on the base completeness criteria of KAOS goal modeling, and would integrate well with a PREview oriented XML notation support for example.

References

[1] Safee project. http://www.respect-it.com/index.php?id=88

[2] Annie I. Anton. Goal-based requirements analysis. In *ICRE '96: Proceedings of the 2nd International Conference on Requirements Engineering (ICRE '96)*, pages 136–144, Washington, DC, USA, 1996. IEEE Computer Society.

[3] John W. Brackett. Software requirements. Technical Report SEI-CM-19-1.2, Software Engineering Institute, Carnegie Mellon University, USA, 1990.

[4] Wael Chatila. W ajax design pattern.

http://waelchatila.com/2006/07/08/1152426781706.html

[5] R. Chitchyan, A. Rashid, P. Sawyer, A. Garcia, Mónica P. Alarcon, J. Bakker, B. Tekinerdogan, S. Clarke, and A. Jackson. A survey of analysis and design approaches. Technical Report AOSD-Europe Deliverable D11, AOSD-Europe-ULANC-9, Lancaster University, May 2005.

[6] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, Boston, USA, 2000.

[7] A. Finkelstein, J. Kramer, and M. Goedicke. Viewpoint oriented software development. In *Proc. of Third Int. Workshop on Software Engineering and its Applications*, pages 337–351, 1990.

[8] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering*, 2(1):31–58, 1992.

[9] A. Finkelstein and I. Sommerville. The viewpoints faq. *BCS/IEE Software Engineering Journal*, 11(1):5–18, 1996.

[10] Eelke Folmer and Jan Bosch. Usability patterns in software architecture. In *Proceedings of the 10th International Conference on Human-Computer Interaction*, volume 1, pages 93–97, Crete, Greece, June 2003.

[11] G. Kotonya and I. Sommerville. Viewpoints for requirements definition. *Software Engineering Journal*, 7(6):375–387, Nov 1992.

[12] G. Kotonya and I. Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1):5–18, 1996.

[13] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Fifth IEEE International Symposium on Requirements Engineering (RE'01)*, pages 249–262, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[14] J. C. S. P. Leite. Viewpoint analysis: a case study. In *IWSSD '89: Proceedings of the 5th international workshop on Software specification and design*, pages 111–119, New York, NY, USA, 1989. ACM.

[15] L. Liu and E. Yu. From requirements to architectural design - using goals and scenarios. In *ICSE-2001 Workshop: From Software Requirements to Architectures*, pages 22–30, May 2001.

[16] A. G Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[17] G. P. Mullery. Core - a method for controlled requirement specification. In *ICSE '79: Proceedings of the 4th international conference on Software engineering*, pages 126–135, Piscataway, NJ, USA, 1979. IEEE Press.

[18] Ken Orr. *Structured Requirements Definition.* Topeka, Kansas, 1981.

[19] Ana Sofia Penim and Manuel Pimenta. Ontologies in software engineering: A kaos ontology. Semantic Web Course, June 2008.

[20] Awais Rashid, Ana Moreira, and Joao Araújo. Modularisation and composition of aspectual requirements. In *AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development*, pages 11–20, New York, NY, USA, 2003. ACM.

[21] Respect-IT. *A KAOS Tutorial*, Oct.18 2007. V1.0.

[22] Douglas T. Ross. Applications and extensions of sadt. *IEEE Computer*, 18(4):25–34, 1985.

[23] I. Sommerville. *Software Engineering*. Pearson Education Limited, Essex, England, 8th edition, 2007.

[24] I. Sommerville and P. Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[25] I. Sommerville and P. Sawyer. Viewpoints: Principles, problems and a practical approach to requirements engineering. *Annals of Software Engineering*, 3:101–130, 1997.

[26] Eric Siu-Kwong Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, Toronto, Ont., Canada, Canada, 1995.

Appendixes

Appendix A – Administrator Viewpoint

Table 8-0-1 Administrator Viewpoint Tabular Representation

Name	Administrator (Admin)
Type	Stakeholder
Focus	The top-ranked user of the application. Manages every aspect of the application and the information it handles.
Requirements	1. In order to manage the hierarchical structure… a. The Admin is expected to add organizational nodes. b. The Admin is expected to eliminate organizational nodes. c. The Admin is expected to edit organizational nodes and in order to do so… i. The Admin is expected to edit organizational node details. ii. The Admin is expected to move organizational nodes. iii. The Admin is expected to manage billable components associations and in order to do so… 1. The Admin is expected to associate component. 2. The Admin is expected to edit association percentage. 3. The Admin is expected to remove association. 2. In order to manage reports automation… a. The Admin is expected to create reports automations and in order to do so… i. The Admin is expected to define a report template. ii. The Admin is expected to define a schedule. iii. The Admin is expected to select destinations.

	b. The Admin is expected to activate reports automation. c. The Admin is expected to edit reports automations and in order to do so… i. The Admin is expected to edit report template. ii. The Admin is expected to edit report schedule. iii. The Admin is expected to edit report destinations. 3. In order to manage assets… a. The Admin is expected to manage asset types and in order to do so… i. The Admin is expected to define an asset type. ii. The Admin is expected to modify an asset cost/type. iii. The Admin is expected to eliminate an asset type. b. The Admin is expected to add an asset. c. The Admin is expected to delete an asset. d. The Admin is expected to edit an asset's details. e. The Admin is expected to manage asset associations and in order to do so… i. The Admin is expected to add asset associations. ii. The Admin is expected to modify an asset association. iii. The Admin is expected to eliminate an asset association.
Sources	Target administrators of the application, accounting department, tech-support.
History	None