A Survey of Requirements Engineering Methods for Pervasive Services

Freeband A-MUSE Deliverable D5.7

Colophon

Date :	March 2006
Version :	Final
Change :	Processed reviewer comments
Project reference:	A-MUSE
Freeband reference :	Freeband/A-MUSE/D5.7
Company reference :	[Company reference]
URL :	[URL]
Access permissions :	Public
Status :	Final
Editor :	P. van Eck
Company :	CTIT
Author(s) :	L. Kolos-Mazuryk, P. van Eck, R. Wieringa

Synopsis:

Comparison of three goal-oriented requirements engineering methods (GBRAM, KAOS and I) from the perspective of pervasive services and context modelling.*

Abstract

Designing and deploying ubiquitous computing systems, such as those delivering large-scale mobile services, still requires large-scale investments in both development effort as well as infrastructure costs. Therefore, in order to develop the right system, the design process merits a thorough investigation of the wishes of the foreseen user base. Such investigations are studied in the area of requirements engineering (RE). In this report, we describe and compare three requirements engineering methods that belong to one specific form of RE, namely Goal-Oriented Requirements Engineering. By mapping these methods to a common framework, we assess their applicability in the field of ubiquitous computing systems.

Preface

The A-MUSE project is part of the Freeband Communication programme, which aims at the generation of public knowledge in advanced telecommunication (technology and applications). Freeband is based on the vision of 4G networks and services. It specifically aims at establishing, maintaining and reinforcing the Dutch knowledge position at the international forefront of scientific and technological developments, addressing the most urgent needs for research and novel applications in the present unfolding of new technology. Freeband comprises more than 25 organisations, including all-important technology providers and many representative end-user organisations. The Dutch Ministry of Economic Affairs is co-funding this programme as part of the BSIK plan

The vision for Freeband for 2010 is to consider communication and information transfer from the perspective of the user, not the provider. The communication infrastructure will become transparent and abundant in all its layers. Freeband addresses the knowledge chain in communication in the direction of the new ubiquitous communication paradigm. Based on this vision key research questions take place in three main themes:

- **Society, Users and Applications:** what are the new possibilities in different sectors for ubiquitous communication and ambient intelligence, what do they presuppose as knowledge and how can they be realised?

- **Networking, Service Provisioning and Generic User Interaction:** the telecommunication infrastructure viewed from the user's perspective.

- **Enabling Technologies:** no new services emerge without adequate technology; conversely, it is the technology that drives the new paradigms!

The A-MUSE project positions itself mainly in the 2nd theme. The goal of the A-MUSE project is to develop knowledge, technology and tools that will help master complexity, facilitate re-use and properly address Freeband characteristics during service design and provisioning. Important Freeband characteristics that are explicitly considered are mobility, context-awareness, attentiveness and personalization. An important implication of the Freeband vision, namely the ability of the user to create personal services and the ability of the system (services infrastructure) to support run-time service creation, is also considered in the project.

The A-MUSE project combines a number of developments that can contribute to the Freeband vision on advanced service design and deployment support: model-driven design, as a way to raise the level of abstraction at which services are specified; service-oriented computing as a new paradigm for distributed

computing; and semantic services, as a solution for making service more ‘intelligent’. These developments have recently drawn substantial research interest, and produce an impact on current standardization.

Table of Contents

1 Introduction

Pervasive (or ubiquitous) computing [1] brings new paradigms for computing models. The idea is to embed networked computing devices in the human environment and to provide various services to the people using these devices. In such a model, the entire world around us is a network of intelligent computing devices, which constantly observe its state. One special instance of pervasive computing are pervasive mobile services, where the presence of fourth-generation mobile networks is the driving force behind the development of mobile, context-aware, personalized services offered to a mass-market of end consumers. It is expected that these services will blend into the lives of these customers just like computing devices will in pervasive computing.

Such services possess a number of specific properties, which are not common for traditional information systems. These properties have to be adequately addressed during the whole design process of such services, including the early stages such as requirements engineering. We feel that existing methods for RE [15][16][17][18][19] are not completely suited for pervasive service development, and new methods have to be researched. The goal of this survey is first of all, to get enough evidence for the need of new methods, and second, to get an overview of existing methods, and underline their advantages and disadvantages in the context of development of pervasive mobile services. Such an analysis gives us a fundament for the development of a requirements engineering method best suited for pervasive mobile services.

We think that the first step in the development of new RE methods for pervasive systems should be development of context modelling techniques. The core of the context model should consist of:

- *A number of context attributes which are relevant for the service;*
- *Possible changes of the values of these attributes, and reaction of the service to such changes;*
- *Activities which cause attribute changes;*
- *Actors which perform the activities;*
- *Context changes can correlate with each other, and this correlation, most probably, will affect desired system behaviour. So, another important part of the context modelling is a study of various interdependencies between context attributes.*

This opinion is a result of the analysis presented in this report of, on the one hand, the specific features of pervasive systems and, on the other hand, the ability of existing requirements engineering approaches to accommodate them.

This report has the following structure. In chapter 2 we present characteristics of pervasive services. Chapter 3 gives an overview of goal oriented requirements engineering and introduces a running example in the area of personal healthcare. The following three chapters give details of the studied GORE

methods—GBRAM, KAOS and I* respectively—and apply them to the running example. The last chapter provides an overview of the results of the comparison and presents our conclusions.

2 Pervasive Services and Their Properties

The goal of pervasive computing is to create an environment where the connectivity of devices is embedded in such a way that this connectivity is unobtrusive and always available. The main idea is that a pervasive service observes its environment (also known as context) and, more importantly, takes decisions based on these observations. For example, a modern car has a lot of sensors and measures quite a lot of things inside the car, such as oil level, tire pressure, water temperature, as well as outside the car, such as outside temperature. However, it cannot be considered a pervasive system, since it does nothing with most of these measurements, except presenting them to the driver. Decision making is completely up to the user of the system, while a pervasive system is capable of making decisions itself.

Before looking at existing requirements engineering methods from the pervasive service perspective, we have to analyze pervasive services to determine which of their specific features have an influence on requirements engineering. We call these specific features 'distinguishing properties' of pervasive services.

It is recognized that one of the most important properties of a pervasive service is context-awareness [2], which means that a system is able to observe its environment using some sort of sensor, that it is aware of this environment and is able to react to changes in it [3][4][5]. Due to the high importance of context-awareness, we divide the other properties of pervasive services in two groups: non-contextual and contextual. In the literature, several properties of pervasive systems are mentioned [2][3][4][8]. Table 1 summarizes them.

#	PROPERTY CLASS	PROPERTY
N1	Non-contextual	Users pay limited attention to an application over a long period of time
N2	Non-contextual	User activity occurs in spurts
N3	Non-contextual	Short time to market
N4	Non-contextual	No common UI standard
N5	Non-contextual	Constrained computational resources
N6	Non-contextual	Always available for use
C1	Contextual	Dynamic system environment
C2	Contextual	Variable bandwidth
C3	Contextual	Changing display characteristics
C4	Contextual	Changing user environment, such as displays, input devices (laptop, PDA, smart-phone)

#	PROPERTY CLASS	PROPERTY
C5	Contextual	Dynamic adaptation to a new environment

Table 1: Distinguishing properties of pervasive systems

Interestingly, even at a first glance one can see a number of design challenges which are posed by combinations of the properties mentioned. For example, property N6 implies that a pervasive system should be always on and running. However, interaction with the user does not occur very often according to N1. And when the user interacts with the system, usually it results in heavy computation and/or data transmission activities (see N2). But we have limited computational resources (including power supply) – so it is a challenge to keep the system always available to the user, preserve power and rationally use scarce computational resources during the user activity spurts. The same holds for contextual properties. As properties C1 – C4 describe the changes that may occur around the pervasive system, property C5 says what a system should do as certain changes are detected. Actually property C5 is the core of context awareness. Getting back to our car example, imagine we have installed a light sensor and extra software. Now the car would be able to switch the lights on/off depending on the outside lighting level, which is an example of context-aware behaviour, typical for pervasive systems. The design challenge here is to distinguish important changes from the ones that can be neglected, detect them and design an adequate reaction strategy. Clearly, these properties result in requirements which are not common for traditional information systems.

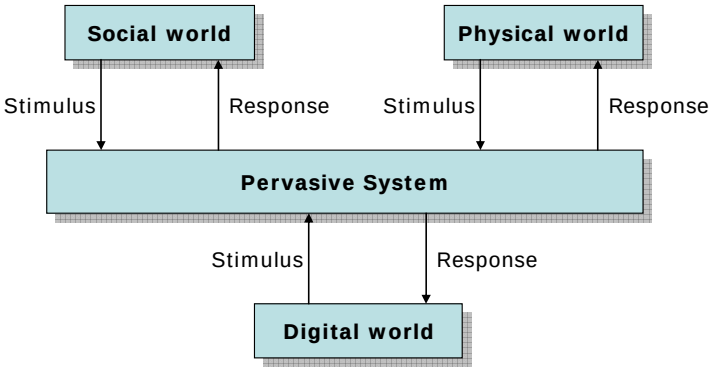

Figure 1: A pervasive system and its environment

So far we used the terms context and contextual rather often, without specifying a precise meaning of these. To avoid further confusion, we would like to explain what we mean by context in our work. Usually a world is represented by three main constituents (see Figure 1):

- *Physical world – everything that is described by physical theories – such as thermodynamics, optics, etc.*
- *Digital world – software components and applications, logic, information resources.*
- *Social world – mental state, tasks, business processes.*

A pervasive system interacts with its environment. The phenomenon of stimulus-response behaviour is described in [6]. Specific to pervasive systems is that response depends on stimuli and on the state of the system context.

We have studied [7] a number of context taxonomies [8][9][10][11][12][13] which provided more detailed refinement of such a model and we have concluded that the most complete taxonomy is the one proposed by Krogstie [8]. Krogstie's taxonomy is summarized in Table 2.

#	CONTEXT TYPE	SUMMARY
1	Spatio-temporal	"Describes aspects related to the time and space. Attributes - time, location, speed, direction, etc."
2	Environment	"Captures the entities that surround the user. Attributes –services, temperature, noise, networks, etc."
3	Personal	"Describes the user state. Consists of physiological and mental contexts. Attributes – blood pressure, pulse, weight, mood, expertise, etc."
4	Task	"Describes what the user is doing. Attributes – goals, tasks and actions"
5	Social	"Describes social aspects of the user context. Attributes – information about friends, relatives, role in society. For example, social context at home is different from social context at work. "
6	Information	"Global and personal information, software, databases"

Table 2: Context Taxonomy (based on [8])

In combination with this taxonomy, contextual properties of a pervasive system gain concrete meaning. So, if for each type of context we define a number of attributes we find important, we can precisely say which of these attributes are changing, and what should be the reaction of the system when such changes are detected.

Next, we will use the distinguishing properties of a pervasive system mentioned earlier (Table 1) together with Krogstie's context taxonomy to evaluate the applicability of the requirements engineering methods chosen for this report to pervasive systems. In our running examples, we will analyze the reflection of these properties in the model of the system obtained by applying each of the RE methods.

Besides the ability of representing the properties stated above and the dimensions of pervasive systems and their context, a methodology should provide a technique to identify them. So, it has to be able to deal with the following problems:

- *Identification and representation of goals of the system and its stakeholders;*
- *Identification and representation of relevant context attributes for a service;*
- *Identification and representation of service's interaction with the user (control actions).*

Together with the distinguishing properties and Krogstie's context taxonomy, the ability of a methodology to address the issues mentioned above form the multidimensional comparison framework of goal-oriented requirements engineering methods that is used in this report.

3 Goal Oriented Requirements Engineering

Goal-oriented requirements engineering (GORE) [15] is concerned with the use of goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements. The choice of goal-oriented requirements engineering approach is explained rather easily. It abstracts from the desired properties of the software system being developed – so it allows us to reason about the requirements even if we do not know these properties. Besides, we think that goal-oriented methods can serve as a good basis for service development for a number of reasons. First of all, GORE methods study the system under development in its context which is essential for pervasive services as well. Secondly, GORE methods possess a number of mechanisms which can help reflecting the properties of pervasive services mentioned above. This, in fact, follows from the ability to view the system in its environment.

Goal-oriented methods provide specific support for coping with high-level user and stakeholder goals, facilitating the exploration of design alternatives and the definition of requirements at a suitable level of abstraction [14].

We have studied a number of goal-oriented methods. KAOS [16] aims at supporting the whole process of requirements engineering – starting from the definition of high-level goals which have to be achieved, up to the requirements, objects and operations which are assigned to the various agents in the system. I* [17] is a framework for representing and organizing knowledge and for supporting reasoning in early stage requirements engineering. GBRAM [18] focuses on the initial identification and abstraction of goals from all available sources of information, regardless of the scope of the knowledge base.

In the following sections we give a short description of each of the studied GORE methods, and analyze how the available mechanisms within a certain method allow the modelling of the system context. As the basis for the context model we take Krogstie's taxonomy. Thus we look at the types of context and context attributes mentioned in Table 2. Besides, we look how changes of context attributes can be accounted for in the goal models used by GORE methods.

3.1 Running example

For a better understanding of the methods we have performed a small experiment, based on the following scenario.

3.1.1 Scenario

Today Martijn caught a cold again. In three days he has to go to a conference in India. Usually, his cold is gone after a day or two. But this time his device, based on the measurements of Martijn’s conditions, says

that it is more serious, and advises him to see a doctor. Martijn follows the advice, and goes to see his doctor. After explaining the situation, he asks for a medicine which will cure him before he leaves for India. The doctor prescribes a medicine and also gives him advice to eat more of certain vitamins and microelements. Prescription and advice instead of being written on a piece of paper is stored in Martijn's health assistant. The Health assistant analyses the prescription immediately and warns Martijn that the prescribed medicine can have a side effects for Martijn's liver, and, proposes a list of possible substitutes. The doctor checks the list, selects several medicines, and changes his prescription immediately. The health assistant approves the prescription, and tries to find the closest drugstore. Unfortunately, the ones that are closest to Martijn's home and the doctor's office do not have the necessary medicines available. The health assistant recommends Martijn to go the drugstore in the city centre and tells him that a bus to the city will leave from the bus-stop near doctor's office in 15 minutes. It also sends the prescription to the drugstore. When Martijn comes there, his medicines are ready and waiting for being picked up. After two days, he is feeling well again and ready to go to India.

After the first day of the conference, Martijn is feeling a bit tired. He thinks that he has not recovered from the long flight yet, and the climate is very different from the Netherlands. His health assistant reminds him to take his medicines and take a nap for an hour or two. Martijn follows the advice, and after a short nap he decided to take a walk in the city and have dinner out since he is fond of Indian food. After coming back from the city, Martijn receives a message from his health assistant. According to its measurements, Martijn's liver is at risk, probably because of the spicy food he had for dinner. Knowing that his liver is Martijn's weak point, the health assistant picks a few medicines which will keep Martijn's liver functioning normally. The health assistant finds out the names of these medicines as used in India, and it turns out that they cannot be purchased without a prescription. So, the health assistant finds a free hour in Martijn's agenda and makes an appointment for him at that time with a doctor in the hotel. The next day, Martijn gets a prescription and his medicines, so he will not have any liver problems after a week of spicy Indian food.

3.1.2 Service decomposition

In order to asses the suitability of the selected goal-oriented methods for service-oriented systems, it would be helpful if we have a vague idea of the desired services of the system we are going to build. Such a view can be obtained from a description of the system given by one of the stakeholders.

Without doubts, the main stakeholder of the digital health assistant is the patient. Let's take his place and look what the patient's view of the system is.

One of the main functions of the device is maintenance of the patient's health condition. It is responsible for measurement and control of the patient's condition and maintenance of patient history and for informing

the patient about the results of interaction with doctors, drugstores and hospitals. The assistance function of the device comes into play when the measured health condition deviates from the desired one. In that case, the health assistant advices what has to be performed in order to bring the health condition back to normal.

If the health assistant is not capable of finding a solution itself, a doctor has to be consulted. However, in order to make use of consultancy, an appointment has to be made with the doctor. This is done by the health assistant. However, it requires access to the doctor's agenda to be granted to the health assistant.

As a result of interaction with a doctor, it is possible that medicines need to be ordered from the drugstore. For that, the health assistant has to check, using a medicine information service, whether it is safe for the patient to use the medicines prescribed (to avoid side effects and undesired interactions between medicines). If not a substitute has to be found, possibly with the help of a doctor. After the check-up medicines can be ordered using a medicine provision service.

4 GBRAM

The Goal-Based Requirements Analysis Method (GBRAM) [18] focuses on the initial identification and abstraction of goals from all available sources of information, regardless of the scope of the knowledge base. As the author claims, a significant difference of GBRAM from other existing goal-oriented methodologies is that GBRAM addresses initial identification and origin of goals, while existing methodologies assume existence of some basic goal documentation.

Roughly GBRAM consists of two types of activities – goal analysis activities and goal refinement activities. Key concepts in GBRAM are goals, stakeholders – those who have interests in completion of identified goals, and agents – those who are responsible for completion of the goals. These are identified during goal analysis processes and elaborated in the refinement processes.

GBRAM goal analysis comprises the following activities:

- *Exploration – extraction and identification of all goals from existing sources of information;*
- *Identification – initial specification of goals and preparation for future elaboration;*
- *Organizing – classification of goals and identification of interdependencies between goals.*

The result of the goal analysis is an organized goal set, which is ready for further refinement and optimization.

Goal refinement concerns three specific activities:

- *Refinement of the goal set to prune its size – elimination of redundancies and reconciliation of synonymous goals;*
- *Elaboration of the goals to uncover hidden goals and requirements – uncovering of hidden goals and requirements;*
- *Operationalisation of the goals into operational requirements.*

Generally, the GBRAM method provides a very elaborate and detailed framework for the definition and elaboration of goals, although some ideas and conclusions are questionable. However, transition from goals to requirements is ill-defined. It might be a good idea to combine GBRAM with one of the requirements analysis methodologies which assume that goals are given. According to the claims of GBRAM authors, there exist a significant number of such methodologies.

GBRAM can be viewed as a goal-centric method, where the goal hierarchy is the main focus, and the rest of the concepts, such as stakeholders and agents, are auxiliary. The fact that agents are identified based on the initial set of goals proves the above statement. In our opinion, the context of the system has a significant influence on the goals, but not the other way around. Logically, the cause has to follow the

effect, but not the other way around. From the above we conclude that GBRAM does not pay enough attention to the system environment.

Indeed, GBRAM does not provide any specific mechanisms for explicit modelling of system's environment, which is a serious disadvantage for the design of pervasive services, since, as mentioned above, the main idea of such services is to be built in the environment on various levels (see context taxonomy) and to react on changes in this environment. Without an adequate service context model it would be a difficult task to analyze what types of context are relevant for the service, select context attributes which have to be monitored, and, finally, to get an adequate set of requirements.

If we look at GBRAM from Krogstie's context taxonomy's point of view, we can easily see that GBRAM operates in the domain of task context as defined by Krogstie. Indeed, the main concepts of GBRAM are goals, with auxiliary concepts of actions, tasks and agents. Other types of context have to be modelled using these concepts, which is cumbersome. Moreover, in most cases the GBRAM model is on a higher abstraction level than the model of context. For example, in the car where the light switches on automatically when it's dark outside the GBRAM model may contain two goals – first, to maintain a certain level of visibility for the driver and, second, to use resources rationally(light bulbs and power). The context model in this example would be rather simple – we are talking only about environment context with only one relevant attribute – lighting. There is actually no straightforward way to embed context information in the GBRAM goal hierarchy.

4.1 Running example

A simplified goal hierarchy obtained by the application of the GBRAM method to our running example is presented in Figure 2. This hierarchy, in principle is the same as the one obtained by KAOS (see Section 5). The difference is that GBRAM gives more insight into the relations between the goals on the same level ('and' and 'or' relationships). On the other side, the approach used in the KAOS method is much more formal than GBRAM, and, thus it allows much more opportunities for verification and validation of the obtained results, and it is more precise as well.

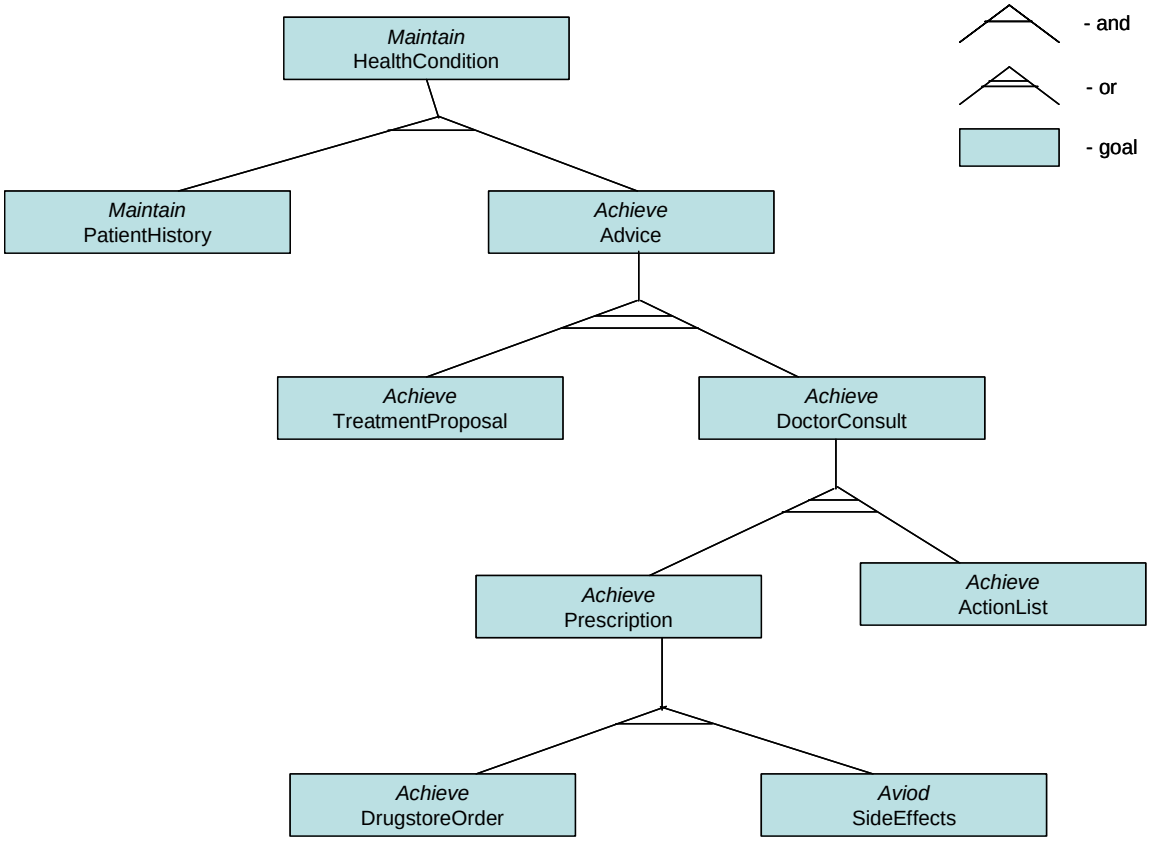

Figure 2: GBRAM Goal Hierarchy

5 KAOS

The KAOS [16] method aims at supporting the whole process of requirements engineering – starting from the definition of high-level goals which have to be achieved up to the requirements, objects and operations which are assigned to the various agents in a system.

The KAOS methodology has three constituents:

- *A specification language;*
- *An elaboration method;*
- *Meta-level knowledge.*

The specification language provides language constructs for capturing the types of concepts that appear in the process of requirements elaboration. The most important concepts are those of an object, operation, agent, goal requisite, etc.

The KAOS language has a two-level structure: an outer semantic net layer for the declaration of concepts, and an inner formal assertion layer for the formal definition of these concepts.

The KAOS elaboration method consists of the following steps:

- *Goal elaboration: elaboration of the AND/OR graph by defining goals and their refinements until assignable requisites are reached. This process is in general a combination of top-down and bottom-up approaches. The source for the goals identification is interviews, analysis of the available documentation, obstacle analysis, scenario-based elicitation, etc;*
- *Object capture: identification of the objects involved in goals formulations, definition of their relationships and of their properties;*
- *Operation capture: identification of object state transitions, which are meaningful to the goals;*
- *Operationalisation: derivation strengthening on operation pre- and post- conditions in order to ensure that all requisites are met. This process is supported by a set of formal derivation rules.*
- *Responsibility assignment: identification of alternative responsibilities for requisites.*

These steps can be run concurrently with possible backtracking at every step. It is important to emphasize the two-level structure of the KAOS language and the complementarities of its levels. At the semantic level, a requirements model is built in terms of concepts described by informal attributes. At the optional formal level, the requirements model is made more precise. Problems coming from informality can be solved and formal reasoning can be used. The formal level can be the basis for the validation of the requirements model. However, as practical experience shows, industrial users prefer to stay on the informal level [16].

With respect to the context modelling KAOS is similar to GBRAM. The main concepts in KAOS – goals, operations, tasks, actions allow adequate modelling of task context of the system. As for the other context types as defined by Krogstie, no modelling mechanisms are available in KAOS. All the differences between KAOS and GBRAM occur at the goal-oriented model level. KAOS is more formal, and, thus, provides better mechanisms for verification and validation of the requirements obtained.

5.1 Running example

Goal Hierarchy

The first step of KAOS is the elaboration of a goal structure. Figure 3 depicts a goal hierarchy for the selected scenarios. The nodes of the hierarchy tree are the goals to be achieved. Every goal is further refined by specifying the subgoals which have to be achieved in order to enable the achievement of the main goal.

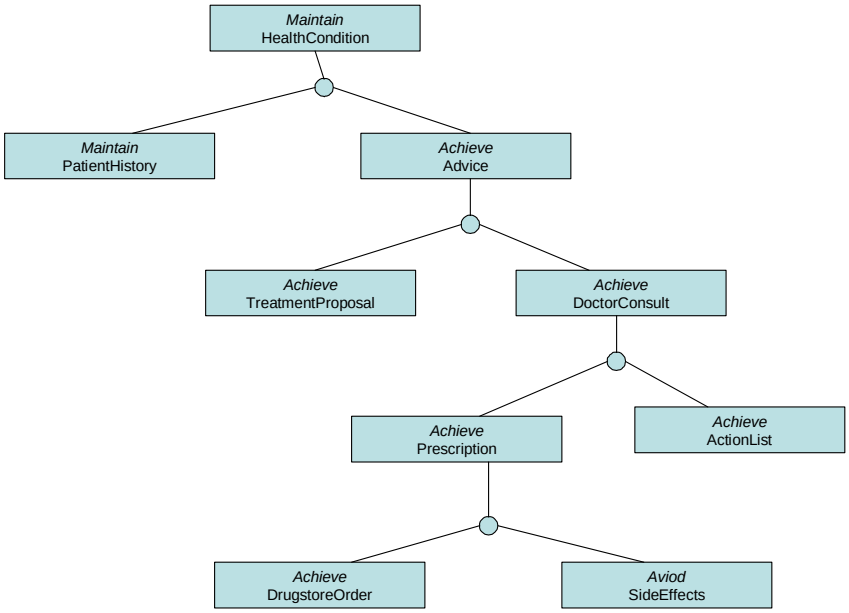

Figure 3: KAOS Goal Hierarchy

Operations and Agents

Based on the results obtained in the pervious stages, we have identified a number of operations and agents capable of performing these operations. Detailed description of how these steps are performed is omitted here, since these are not the main focus of our research and are given in details in the KAOS literature. Table 3 provides a summary of the operations identified.

OPERATION	AGENTS
Measure condition parameters	Health Assistant (HA), Doctor
Analyze condition parameters	HA, Doctor
Record measurement into treatment history	HA
Give and advice for correction of condition parameter if deviation is not big	HA, Doctor
Consult medicineDB for possible treatments	HA
Make an appointment with doctor	HA
Receive action list from doctor	Patient, HA
Receive prescription from doctor	Patient, HA
Analyze received prescription for side effects or conflicting medicines	HA, Doctor
Check treatment history	Doctor, HA
Find the nearest drugstore with needed medicines available	HA, Patient
Submit an order to the drugstore	HA, Patient

Table 3: Operations and Agents

6 I*

I* is a framework for representing and organizing knowledge and for supporting reasoning in the early stage of requirements engineering. It consists of two main modelling components: the Strategic Dependency Model and the Strategic Rationale Model. The Strategic Dependency model [17] is used to describe dependencies between the actors in the organizational context. It shows the external relationships between actors, while hiding the intentional constructs within each actor. This model can be useful in helping to understand organizational and system configurations – both existing and proposed ones. The Strategic Rationale model is used to describe stakeholder interests and concerns, and how they can be addressed by various system configurations. In fact, it provides a way of modelling stakeholder interests, and how they might be met, as well as stakeholders' evaluations of various alternatives with respect to their interests.

The key concept in I* is that of an actor. Actors have intentional properties such as goals, beliefs, abilities and commitments. Actors depend on each other for the goals to be achieved, tasks to be performed, and resources to be furnished.

The following are the most significant advantages of using the I* methodology [19]:

Knowledge representation and reasoning – I* provides a simple informal way to represent knowledge about actors and their dependencies, therefore assisting in goal identification.

Degree of formality – early stages of RE are likely to be highly interactive, with the stakeholders as the primary source for information. A scheme for knowledge representation can facilitate knowledge management and reasoning and ensures better understanding between stakeholders and developers.

Incorporating intentionality – introduction of intentional concepts, such as goals, facilitates reasoning and understanding of why the system is needed.

Multi-lateral intentional relationships – it is essential to describe how the system depends on its environment, as well as how the environment depends on the system.

However, the goal model of I* is very basic, and the goals are not refined or elaborated. The I* methodology obviously focuses on the early stages of requirements engineering and does not describe the transition from stakeholder goals to systems requirements.

Compared to GBRAM and KAOS, which state the importance of goal identification, I* gives a much more elaborated framework for goal identification, while the former two approach the problem quite simplistically.

On the contrary, I* is not focusing on the refinement of obtained goals. Although the I* methodology is not very elaborate as a whole, it has very wide modelling capabilities, which of course, allow much better modelling of context than KAOS and GBRAM do.

First of all, I* sees the system in its environment. This is an important first step in modelling the system context. The strong points of I* are in social and task modelling, and partly in environment and information context modelling. Social context can be modelled with such concepts as agents, dependencies and relationships. For modelling task context I* possesses the concepts of tasks and goals. Certain attributes of the environment and information context can be modelled as I* resources.

In our opinion, the strongest mechanisms of I* are various types of relationships between the model entities. These allow modelling the system and system context as a whole and lead to a more adequate analysis of those.

With respect to the car example we mentioned before, I* surely provides more possibilities than GBRAM or KAOS. In I*, our example car can be modelled as an actor that possesses the following resources – lights and a power source. The goals of the car are, similar to the goals in the GBRAM and KAOS models, to maintain a certain level of visibility and to use resources efficiently. In I*, these resources appear in the model explicitly, unlike in KAOS and GBRAM. I* allows us to include the environment in the model as an actor, which owns a resource - natural lighting. The car depends on the availability of this resource. Such a model describes more aspects than the ones of KAOS and GBRAM. However, it is not perfect either. For example, we miss a description of the dynamics of such a system. However, in an I* model we can embed some context attributes.

6.1 Running example

The main actors in the I* model of our running example are the patient, the health assistant, doctors and drugstores. The main goal of the patient is maintaining the normal health condition. For reaching that goal, the patient depends on other actors – health assistant, doctor and drugstore. This goal has two sub-goals – getting treatment and getting medicines when needed. The goals of the health assistant are to check the condition of the patient and to give him advice or arrange treatment. Advice can be given based on the information obtained from the following resources – medicine DB and the patient's treatment history. For treatment an appointment with a doctor has to be made, for which that doctor's agenda has to be consulted. During the consultancy, the doctor advises patient what to do to get back to normal, and prescribes some medicines when needed. The task of the health assistant is to check the prescription for the medicines which cannot be taken by the patient. After the subscription was approved, the health assistant submits an order to the nearest drugstore that has the medicines needed available. After that patient can receive the medicines.

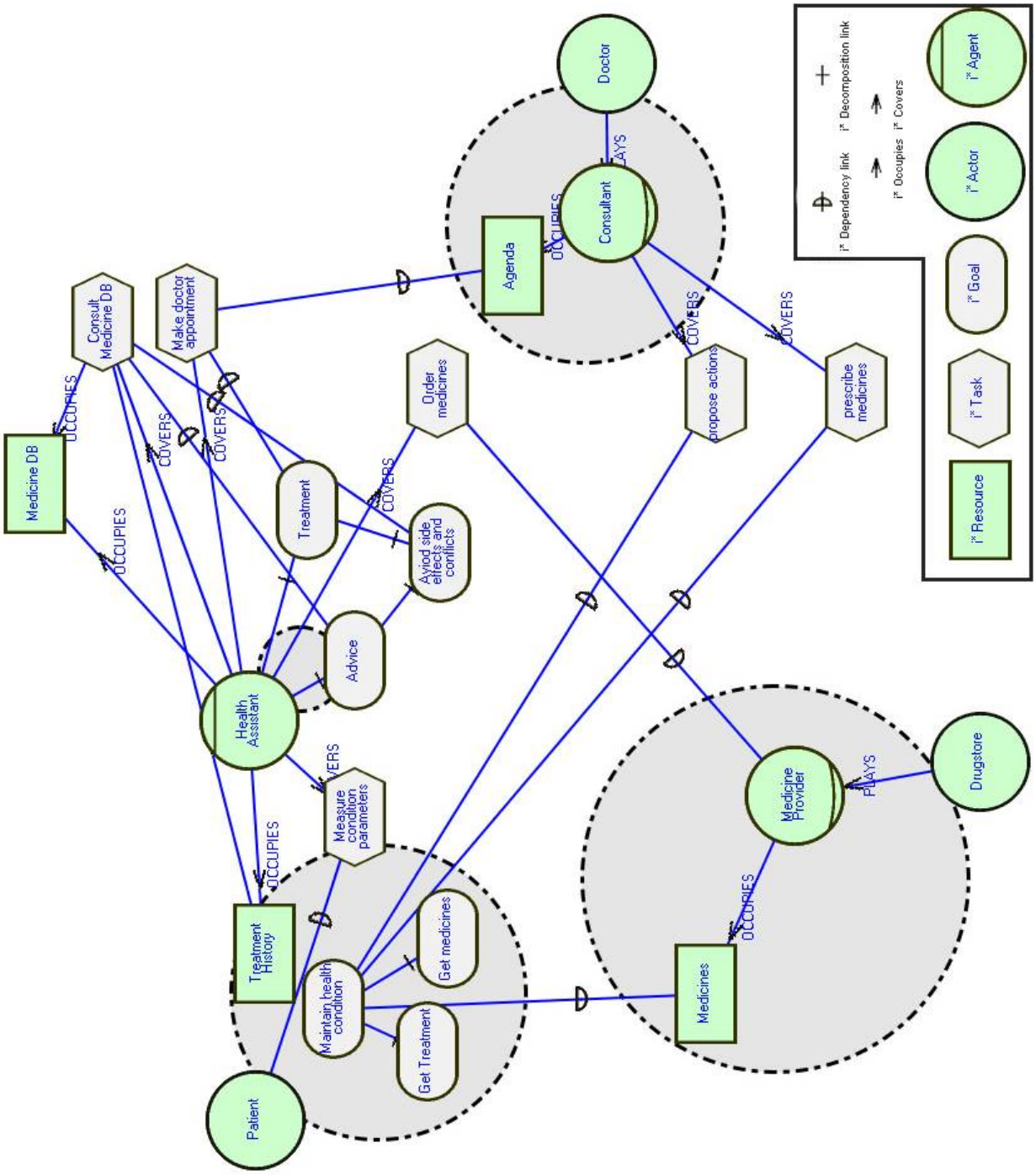

Figure 4: I* Model

7 Analysis and Conclusions

Generally, we have seen that models created by using existing GORE methods are not fully suitable for the development of pervasive services. Although obtaining a set of requirements adequately describing the desired behaviour of a pervasive service using GORE methods is possible, the task is still rather complicated, and requires extra effort from the engineer/designer. The main problem seems to be the absence of mechanisms for modelling the service environment, which is known in the world of pervasive computing as context.

We have shown that capabilities for context modelling of existing GORE methods are not sufficient. GORE methods operate with a limited number of concepts (see Table 4), which do not allow modelling context and relevant context attributes directly. An overview of the expressive power of GORE methods for context modelling is presented in Figure 5.

CONCEPTS	KAOS	GBRAM	I*
Goal	✓	✓	✓
Object	✓		
Operation	✓		
Agent	✓	✓	✓
Requisite	✓		
Goal dependency	✓	✓	✓
Stakeholder	✓	✓	✓
Task			✓
Resource			✓
Decomposition			✓
Relationships			✓

Table 4: Main concepts used in GORE methods

The figure shows per GORE method to which extent the method is able to model context attributes directly using the techniques and concepts available in the method. The bigger the area covered by a polygon is, the better the coverage of context modelling provided by the correspondent method. We would like to

emphasize that the picture is based on our observations and experiences with the methods studied in the previous chapters. GBRAM and KAOS provide mechanisms that can adequately reflect only one of six context types. As mentioned above these methods are pure goal-oriented methods, and do not pay enough attention to the explicit modelling of the system environment. An environment model is the basis of the I* method. However, even I* cannot adequately model spatio-temporal and personal contexts. Information and environment contexts can be represented only partly. Such observations bring us to conclusion that a new approach is needed.

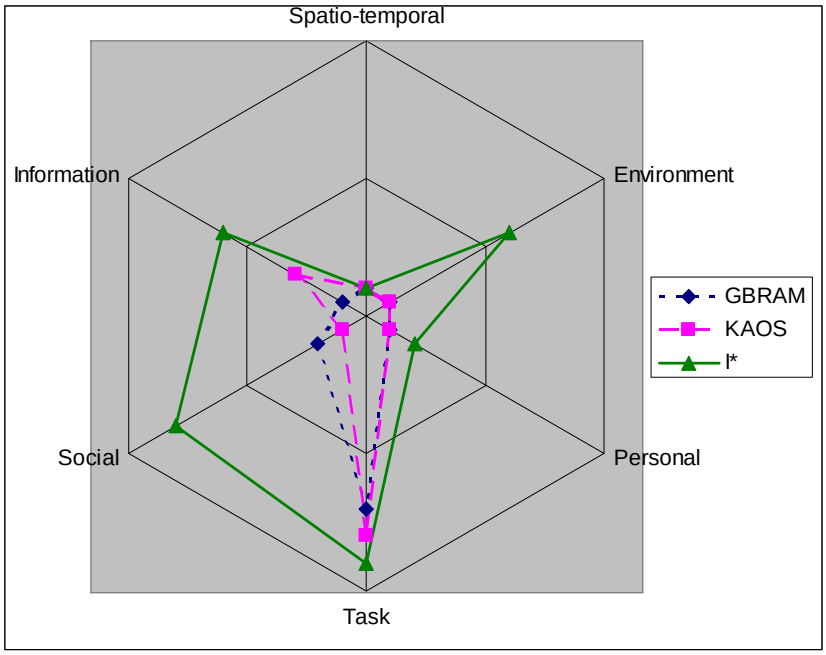

Figure 5: Relative expressive power of GORE methods for context modelling

As we mentioned in the beginning of this report, context modelling is an important part of our comparison framework, but not the only one. Table 5 gives an overview of how the methods studied are able to deal with the main issues in requirements engineering for pervasive services mentioned before.

	KAOS	GBRAM	I*
Goals identification	Covered	Covered	Partly covered
Context attributes identification	Not covered	Not covered	Partly covered
Interaction identification	Partly covered	Partly covered	Partly covered

	KAOS	GBRAM	I*
Goals representation	Covered	Covered	Covered
Context representation	Not covered	Not covered	Partly covered
Interaction representation	Partly covered	Partly covered	Partly covered

Table 5: GORE methods and issues in RE for pervasive services

Goal identification and representation are covered well by GORE methods, since they actually have been designed for that. However, KAOS and GBRAM possess a better approach to goal identification than I*. On the other hand, I* is explicitly studying the system in development in its environment. Therefore, I*, unlike KAOS and GBRAM, has some mechanisms for identifying and representing context attributes. However, it covers only a part of them, as we see on Figure 5. I* does not comprise any formal identification procedure for context attributes either. All the methods partly cover the identification and representation of the interaction between the system and the user. While KAOS and GBRAM have to deal with interaction processes at some level in the goal hierarchy, I* models can also point to interactions in the system as well.

In this report, three methods have been compared from the perspective of context modelling. In our opinion, goals and interactions in pervasive systems have to be well-defined as well, and then approaches to identification and representation of these can be chosen. Since context is essential for pervasive services, and is fairly well defined by Krogstie's taxonomy, we think that the first step in the development of new RE methods for pervasive systems should be development of context-modelling techniques. The core of the context model should consist of:

- *A number of context attributes which are relevant for the service;*
- *Possible changes of the values of these attributes, and the reaction of the service to such changes;*
- *Activities, which cause attribute changes;*
- *Actors, which perform the activities.*
- *Context changes can correlate with each other, and this correlation, most probably, will affect desired system behaviour. So, another important part of the context modelling is the study of various interdependencies between context attributes.*

Further, a context model of a pervasive service has to be connected to its functional model, described by the goals and the interaction with (and within) the system. Evaluating methods from this perspective is left for future research.

References

[1] M.Weiser (1993). Hot Topics: Ubiquitous Computing. *IEEE Computer*, 26(10):71-72, October 1993.

[2] J. Shen and X. Shen (2001). User Requirements in Mobile Systems. *Proceedings of the 2001 Americas Conference on Information Systems*, August 2-5, 2001, pp. 1341-1344.

[3] A.K. Dey and G.D. Abowd (1999). *Towards a better understanding of context and context-awareness*. GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology.

[4] J. Pascoe, N.S. Ryan and D.R. Morse (1999). Issues in developing context-aware computing. In: Gellersen, H.-W. (Ed.) *Handheld and Ubiquitous Computing. First International Symposium, HUC'99*. Karlsruhe, Germany, Sept. 1999. Lecture Notes in Computer Science, vol. 1707, pages 208-221. Springer-Verlag.

[5] B.N. Schilit, N.I. Adams and R. Want, R. (1994). Context-aware computing applications. *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp. 85-90. IEEE Computer Society, Santa Cruz, CA.

[6] R. J. Wieringa (2003). *Design Methods for Reactive Systems.* Morgan Kaufmann Publishers.

[7] L. Kolos-Mazuryk, G.-J. Poulisse and P. van Eck (2005). Requirements Engineering for Pervasive Services. OOPSLA'05 workshop on Creating Software for Pervasive Services. San Diego, USA, 2005.

[8] J. Krogstie (2001). Requirement Engineering for Mobile Information Systems. In: *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ'01)*. Interlaken, Switzerland, 2001.

[9] A. Dix, T. Rodden, N. Davies, J. Trevor, A. Friday, and K. Palfreyman (2000). Exploiting space and location as a design framework for interactive mobile systems. *ACM Transactions on Human Computer Interaction*, 7(3):285–321, September 2000.

[10] G. Chen and D. Kotz (2000). *A Survey of Context-Aware Mobile Computing Research*. Dept. of Computer Science, Dartmouth College, 2000.

[11] A. K. Dey and G. D. Abowd (2000). Towards a Better Understanding of Context and Context-Awareness. In: *Workshop on The What, Who, Where, When, and How of Context-Awareness*, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, April 3, 2000. Also GVU Technical Report GIT-GVU-99-22.

[12] T. Rodden, K. Chervest and N. Davies (1998). Exploiting Context in HCI Design for Mobile Systems. In: *First Workshop on Human Computer Interaction with Mobile Devices*, Glasgow, 21st & 22nd May 1998.

[13] B. Schilit, N. Adams and R. Want (1994). Context-Aware Computing Applications. In: *IEEE Workshop on Mobile Computing Systems and Applications*. pp. 85-90. doi:10.1109/MCSA.1994.512740.

[14] D. Bolchini and J. Mylopoulos (2003). From Task-Oriented to Goal-Oriented Web Requirements Analysis. In: *Fourth International Conference on Web Information Systems Engineering*. Roma, Italy. pp. 166-175. IEEE Computer Society. doi:10.1109/WISE.2003.1254480.

[15] A. van Lamsweerde (2004). Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice. *Proceedings of RE'04, 12th IEEE Joint International Requirements Engineering Conference*, Kyoto, pp. 4-7. IEEE Computer Society.

[16] P. Bertrand, R. Darimont, E. Delor, P. Massonet and A. van Lamsweerde (1998). GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering. In: *Proceedings ICSE'98 - 20th International Conference on Software Engineering*, IEEE-ACM, Kyoto, April 98.

[17] E. Yu (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: *Third IEEE International Symposium on Requirements Engineering (RE'97)*, IEEE Computer Society, pp. 226-235. doi:10.1109/ISRE.1997.566873.

[18] A. Antón (1997). *Goal Identification and Refinement in the Specification of Software-Based Information Systems*. Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, USA.

[19] J. Mylopoulos, L. Chung and E. Yu (1999). From Object-Oriented to Requirements Analysis. *Communications of the ACM,* 42(1):31-37.