Modelling with Problem Frames: Explanations and Context in Ambient Intelligent Systems

Anders Kofod-Petersen[1] and Jörg Cassens[2]

[1] Department of Computer and Information Science,
Norwegian University of Science and Technology,
7491 Trondheim, Norway
anderpe@idi.ntnu.no

[2] Institute for Multimedia and Interactive Systems (IMIS),
University of Lübeck,
23562 Lübeck, Germany
cassens@imis.uni-luebeck.de

Abstract. When designing and implementing real world ambient intelligent systems we are in need of applicable information systems engineering methods. The tools we find in the intelligent systems area focus on the knowledge engineering parts, whereas traditional software engineering techniques are usually not designed with the peculiarities of intelligent systems design in mind. This holds in particular for explanation-aware intelligent systems. This work looks at problem frames for explanations and investigates how problem frames can be used to elicit, analyse, and specify these specific requirements. The point of departure is an existing ambient intelligent information system for the hospital ward domain. The work presented here analyses how such a system can be redesigned with a focus on explanation-awareness.

1 Introduction

Ambient intelligence describes environments where human beings are surrounded by intelligent artefacts supported by computing and network technology. Such environments augment everyday objects such as furniture and clothes. In addition, an ambient intelligent environment should be aware of the presence of a person, perceive the needs of this person, and respond to them in an unobtrusive and intelligent manner [1]. Ambient intelligence is laying in the intersection of pervasive computing, ubiquitous computing, and artificial intelligence.

The ability to explain itself, its reasoning and actions, has been identified as one core capability of any intelligent entity [2]. The question of what is considered to be a good explanation is context dependent [3], leading to the necessity to design the explanatory capabilities of an ambient intelligent system together with the modelling of the different situations the system is likely to encounter.

The work presented in this paper targets the requirements elicitation, analysis, and specification processes. We make use of the notion of problem frames [4], which appears to be a promising method both in helping to elicit requirements and in later transformation of design documents into actual systems. We have previously suggested additional problem frames that target explanatory capabilities explicitly [5], and we will here demonstrate how problem frames can be put to use in revealing limitations of an existing ambient intelligent systems design and can help to take needs into account arising from explanatory capabilities when (re-) designing such a system.

The rest of the paper is organised as follows: Section 2 gives an overview of related work; Section 3 give a short introduction to problem frames; Section 4 details the use of problem frames specific to ambient intelligent systems; Section 5 briefly describes the existing ambient intelligent information system for hospital wards and how *context awareness* and *context sensitivity* is applied; Section 6 describes how the requirements for a redesign of the existing application can be analysed and specified by applying problem frames. The papers ends with a conclusion and outlook on future work.

2 Related Work

The use of patterns [6] is common for different software engineering approaches. Patterns can be used in different software development phases and they can have different foci. We can also identify knowledge engineering approaches making use of patterns for the development of intelligent systems. This includes efforts to provide reusable architectures by describing the abilities of (a library of) generic problem solving methods. An example for a component model is the Unified Problem-Solving Method Development Language UPML, cf. [7].

There are several methods and languages that use patterns and focus explicitly on the knowledge aspects of system design. For example, the goal of the INRECA [8] methodology is to support the development of (industrial) case-based reasoning (CBR) applications. Software process models from existing CBR applications are stored in an experience base that is structured at three levels. The *common generic level* is a collection of very generic processes, products, and methods for CBR applications. At the *cookbook level*, we find software models for particular classes of applications (so called recipes). At the *specific project level*, experiences from particular projects are stored. We can identify the recipes at the cookbook level as patterns.

Another well-known approach is the CommonKADS methodology [9]. It is based on two different views on the development process of knowledge based systems: the *result perspective* encompasses a set of models of different aspects of the knowledge based system and its environment, and the *project management perspective* starts from a spiral life-cycle model that can be adapted to the particular project. The CommonKADS template knowledge model provides a way of (partially) reusing knowledge models in new applications and can be understood as patterns in a software engineering sense.

When we look towards the software engineering world, we can see that patterns are used in different phases of the design process.

On the software architecture level, we find *architecture patterns* [10]. At this level, we encounter concepts like 'Blackboards', 'Model-View-Controller', or

'Pipes and Filters'. For finer grained software development close to the actual implementation, one can make use of design patterns that look inside towards the computer and its software [11]. Design patterns deal with concepts like 'Factories', 'Facade', and 'Decorater'.

Early on in the requirements engineering process, Jackson's *problem frames* [4] are at the core of a method to classify software development problems. Problem frames look out into the world and attempt to describe the problem and its solution in the real world. Problem frames introduce concepts like 'Information Display' and 'Commanded Behaviour'. Jackson's set of basic problem frames can be extended to be better able to model domain specific aspects. For example, Hatebur et al. [12] introduce new problem frames for security problems.

Phalp and Cox demonstrate that people are capable of selecting the right problem frames [13]. This study suggests that problem frames are indeed a suitable method for correctly assigning formal models to a given problem description and therefore a helpful tool in the requirements engineering process.

Hall and Rapanotti [14] have introduced extensions to the basic problem frames that will better facilitate socio-technical systems. They introduce a 'user interaction frame', and employ the model-view-controller perspective to ease decomposition. We will build on this results in our own work on ambient intelligent systems as a special class of socio-technical systems where user interaction is not only achieved via explicit communication, but also through the behaviour of both system and user.

3 Problem Frames

The main purpose of any problem frame [4] is to propose a machine that improves the combined performance of itself and its environment by describing the machine's behaviour in a specification. Jackson originally described five different basic frames. In general, a problem frame assumes a user driven perspective. Most basic frames assume that the user is in control and dictates the behaviour of the machine. Since intelligent systems (ideally) take a much more pro-active approach and mixed initiative issues become relevant, new problem frames addressing these topics have to be developed. For the course of this paper, we will focus on frames targeting explanatory aspects and will not discuss other types.

Problem frames can be described by problem frame diagrams. These diagrams consist basically of dashed ovals, representing the requirements, plain rectangles, denoting application domains, and a rectangle with a double vertical stripe, standing for the machine (or software machine) domain to be developed. These entities become the nodes of the frame diagram. They are connected by edges, representing shared phenomena and denoting an interface. Dashed edges refer to requirement references. Dashed arrows designate constraining requirement references.

The domains are of different types, indicated by a letter in the lower right corner. A 'C' stands for a *causal* domain whose properties include predictable causal relationships among its phenomena. A 'B' denotes a *biddable* domain that lacks positive predictable internal behaviour. Biddable domains are usually associated with user actions. An 'X' marks a *lexical* domain. Such a domain is a physical representation of data and combines causal and symbolic phenomena.

4 Problem Frames and Ambient Intelligence

Following the definition of ambient intelligence [1], in general an ambient intelligent system can be fitted into a *Required Behaviour* problem frame. Figure 1 illustrates this. `AmI!C1` is the phenomena shared between the machine and the environment, and controlled by the systems; that is the actuators. The `E!C2` is the phenomena shared between the machine and the environment, which is not controlled by the machine; that is the sensors. Finally, `C3` refers to the behaviour that the machine is to exhibit.

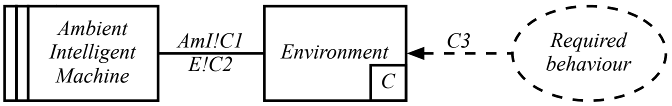

Fig. 1. Ambient Intelligent Systems as a Controlled Behaviour Frame

Even though required behaviour frames are generally and by definition suitable for ambient intelligent systems, some special cases exist where explicit user interaction is required other than through *behavioural interfaces.* It has been argued that any adaptive system in general, and an ambient intelligent system in particular, must be able to change its interaction style and exhibit the ability to explain its behaviour [15]. This ability requires that the system can communicate with the user through suitable interfaces, such as displays. In addition, the user should have the option to explicitly request an explanation of the system's behaviour. Thus, a suitable problem frame is required to capture this.

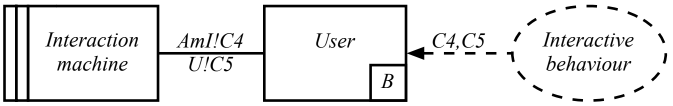

Fig. 2. User Interaction Frame (adopted from [14])

Following the argumentation of Hall and Rapanotti [14], we employ the *User Interaction Frame*, depicted in Figure 2. `AmI!C4` is the symbolic phenomena shared between the machine and the user, where the machine can display information to the user. `U!C5` is the causal shared phenomena between the machine

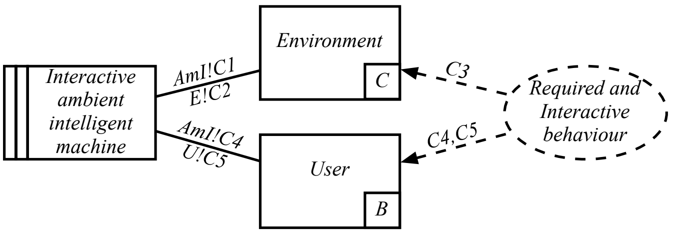
Interactive ambient intelligent machine
AmI!C1
E!C2
AmI!C4
U!C5
Environment
C
User
B
C3
C4,C5
Required and Interactive behaviour

Fig. 3. Interactive Ambient Intelligence Frame

and the user, where the user initiates commands. Finally, `C4,C5` are the rules of conduct between the machine and the user.

Again following Hall and Rapanotti, we can combine these two frames into an *Interactive Ambient Intelligence Frame*, as depicted in Figure 3. Here, interactive, explanatory capabilities are combined with the environment controlling aspects of ambient intelligent systems. This aggregation differs significantly from the original *required behaviour frame* [4]. The behaviour of the ambient intelligent system is not mainly guided by explicit input from the user, but is a result of the pro-activeness of the system and implicit interaction (for example the location of the user). But it opens up for direct interaction, for example by the user requesting an explanation. This will, however, not command the whole behaviour of the system directly, but only a small part of it. In that sense it further on differs from the *user commanded frame* in [14] as the system can take actions that are not triggered through commands explicitly issued by the user.

4.1 Explanation Problem Frames

Taking a closer look at Figure 3, we will see that the upper part captures the behaviour of the ambient intelligent system, whereas the lower part represents the interactive properties of the system. We will use this part of the general frame to model the explanation abilities. To this end, however, we have to decompose the lower part in order to model different types of explanation.

The list of explanation requirements can be described as a list of the explanation goals that a system must be able to satisfy. Sørmo et al. identify five different explanations goals that a system might have to handle [2]. This work has been further expanded in [15], where the explanation goals have been combined with the ambient intelligence paradigm. Our own work focuses on the four goals that are not related to applications as an educational tool: The goal of *transparency* is concerned with the system's ability to explain how an answer was reached. *Justification* deals with the ability to explain why the answer is good. When dealing with the importance of a question asked, *relevance* is the goal that must be satisfied. Finally, *conceptualisation* is the goal that handles the meaning of concepts.

Following these four goals of explanations, we have previously constructed four problem frames that each captures one kind of explanation [5]. The *transparency*

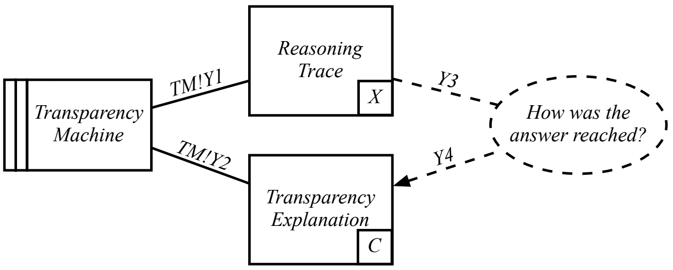

Fig. 4. Transparency Explanation

goal is concerned with how the system finds a solution to a given problem. This allows the user to inspect the reasoning process to identify the cause of any abnormal behaviour. The transparency explanation frame is depicted in Figure 4. Here the `Reasoning Trace` is a lexical domain, which allows the `Transparency Machine` to read the reasoning trace trough the shared phenomena `TM!Y1`. The `Transparency Explanation` is the causal domain, which the machine can control through the shared phenomena `TM!Y2`. In short, the `Transparency Machine` has to inspect the reasoning trace and present the relevant information to its user.

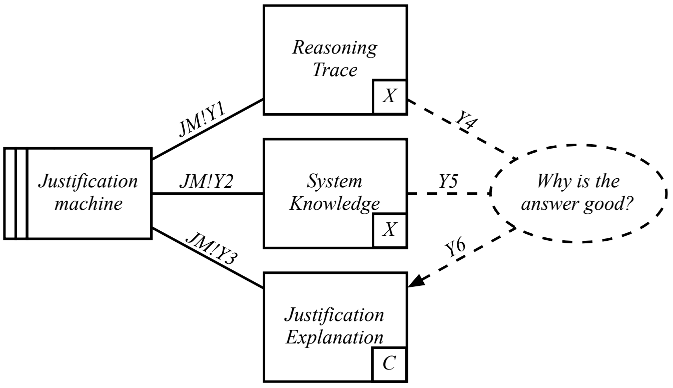

Fig. 5. Justification Explanation

The *justification goal* is closely related to the *transparency goal*. Justification can be seen as a simplification of the reasoning process that the system actually goes through. The main purpose of this explanation goal is to convince the user that the reasoning is sound. In general, whereas transparency explanations are for experts that are interested in the exact details of the reasoning, justification explanations are for novice users that are interested in being persuaded of the system's reasoning. Figure 5 displays the problem frame of a justification explanation goal. This frame resembles the one for transparency explanations, with the addition of the lexical domain `System Knowledge`. This domain facilitates the expansion of a transparency explanation by allowing the `Justification Machine` to inspect the system's knowledge through the shared phenomena `JM!Y2`.

5 Hospital Ward System

The application in question is an ambient intelligent information system for supporting medical personnel at a hospital ward. The persons involved deal with different activities, like ward rounds, pre-ward round meetings, and different forms of examination. The staff has to access a large variety of different information systems. The main goal is to have a system that makes the information sources needed in different situations (such as specific journals, test results, and treatment plans) available pro-actively. To this end, the system must first identify the activity the system's user is involved in, identify his role, and then query the information sources that are likely to be accessed.

Following the definition of ambient intelligence in [1], the system *perceives* its environment, becomes *aware* of ongoing situations, and is *sensitive* to the idiosyncrasies of the particular situations.

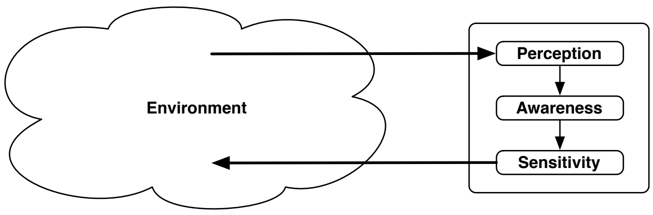

Fig. 6. System Architecture

The existing system is build around a multi-agent platform. *Perception* is handled by a Context Middleware [16], the situation *awareness* is build around the case-based reasoning system CREEK [17], and the acquisition of relevant information (*sensitivity*) is facilitated by dynamic task decomposition [18]. Situations were identified and the knowledge model populated through an ethnographical study conducted at the cardiology ward [19]. The whole system was implemented using the Jade [20] agent framework.

In our system, we use explanations in two distinct ways: first, enhancing and promoting the reasoning process; called the *system centric* view. Second, delivering some knowledge about the reasoning process, its results, or implication to the user; called the *user centric* view. Table 1 shows how these two different

Table 1. Context and Explanations

	Context Awareness	**Context Sensitivity**
System Centric	Explanations to **recognise** the situation	**Identify** the desired system behaviour
User Centric	**Elucidate** why a situation was identified	**Explicate** why a certain behaviour was chosen

usages of explanations relate to the *awareness* and *sensitivity* aspects of our system. For the purpose of this paper we will disregard the perception layer of the architecture as the perception layer demonstrates no reasoning capabilities, and only structures perceived data syntactically.

In the setting of our general frame for interactive ambient intelligent systems depicted in Figure 3, the *system centric* explanations relate the upper part, whereas the *user centric* explanations relate to the lower part. The explanation problem frames for user goals can be put to use in the lower part or *user centric* view, but modified versions reflecting the system's intentions are important for the upper part or *system centric* view as well. For the remainder of this paper, we describe how to make use of explanation problem frames for the explication aspect, describing the user centric and context sensitive use of explanations.

5.1 Example

To clarify the functionality of this system, we will present a small example. It sketches an execution taken from a simulated system run, using the real data set gathered at the cardiology ward. In this case we are dealing with a *pre-ward round* situation. A pre-ward round is a particular type of meeting that occurs every morning. Usually, the physician in charge and the nurse in charge are present. They discuss each of their patients, including their current condition, any changes, and the treatment plan.

The Context Middleware monitors the different sensors in the environment, and discovers a change, which provokes a change in the current context [16]. This change in the context is transferred to the CBR sub-system, which retrieves the best matching case based on the sensor values.

In this example, the CREEK component retrieves a case describing another pre-ward round. Having identified the ongoing situation as a pre-ward round, the CBR engine now extracts the goal of this type of situation. In this case, the goal is to gather the relevant information. This goal is sent to the sensitivity part to be solved [21].

The Sensitivity part of this system receives the goal and matches it to a general decomposition tree that contains the tasks required to satisfy the goal. In this example the task tree that matches a pre-ward round goal is as follows:

1. Acquire name of patient.
2. Acquire changes in patient's conditions since yesterday.

3. Acquire any new results from tests.
4. Examine, and possible change, medication scheme.
5. Note changes in treatment.

Each solvable task is matched to an action performed by an available, willing and able agent. The system currently offers 19 different agents, each representing one information system at the hospital ward. Together these 19 agents offers 21 different information services.

The initial problem of finding the patients name can be facilitated by the *Patient List Agent*. The 'Acquire Information' task is decomposed into one task that acquires changes which are supplied by the *Electronic Patient Record*, the *WiseW* application and the *Patient Chart*, and another task that acquires results which can be delivered by the *Patient Chart* and the *WiseW* application.

This plan is now executed and the information acquired through the different agents is returned to the user; thus ending an execution cycle of this system. Figure 7 depicts the decomposition tree constructed for this example, including the matching agents and their services.

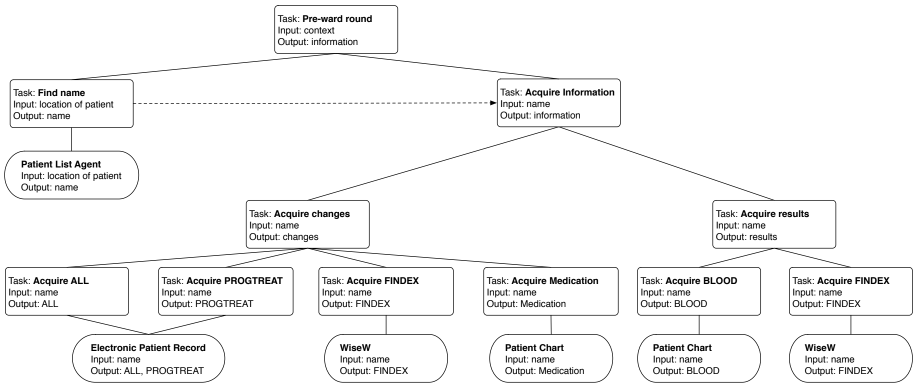

Fig. 7. Pre-ward Round Plan

6 Redesigning the Existing Application

As stated in the previous section, we have performed an ethnographical study to elicit the requirements for an ambient intelligent hospital ward information system. The results of the study were used to model the different situations the system could encounter. Further on, the analysis of artifacts used during the different situations was incorporated into the task decomposer so that suitable information source could be queried in different contexts.

In the first incarnation, explanatory capabilities were not explicitly included in the design specifications. However, the socio-technical theory used in the study design and application allows us to elicit the possible explanation goals users of the system might have [15]. Therefore, a re-design on the grounds of the data already gathered is feasible.

6.1 Example

Revisiting the example of the previous section, we have the instance where the system correctly classifies an ongoing situation as a pre-ward round. If we focus on the context sensitive part of the system, its main purpose is to examine the artifacts, represented by agents, in the environment and find those that can supply relevant information. So far this application only supplies information without any explanation of its behaviour.

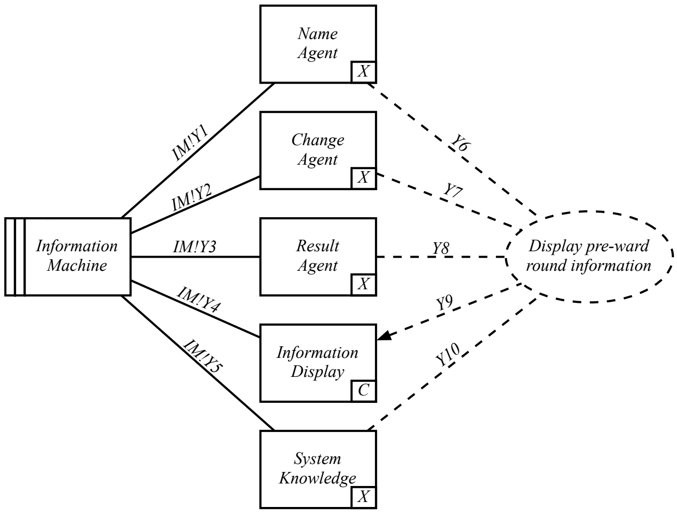

Fig. 8. Ambient Information Display

In order to demonstrate how the explanation goal problem frames can be used to model explanatory needs in the problem domain, we will start with a simplified problem diagram for our application (Figure 8). This is essentially modelling the behaviour of the system without direct user interaction and reflecting the capabilities the existing system was designed to have. This part is a decomposition of the upper part of the *Interactive Ambient Intelligence Frame* from Figure 3. We have modified Jackson's *information display* problem frame and used it as a starting point for the diagram. You can see three domains representing (groups of) the agents mentioned above.

Additionally, you see the `Display` domain which stands for the information display of the system and `System Knowledge` for deciding which data sources to use. For the sake of clarity and simplicity, we have abstracted away the sensor parts of as well as the context aware parts of our example application and focus solely on the information gathering and display parts. Let us now assume that the results displayed by the system are of such a nature that the physician using the system requires an explanation. Let us for the sake of simplicity of the example further focus on a subset of the identified explanation goals.

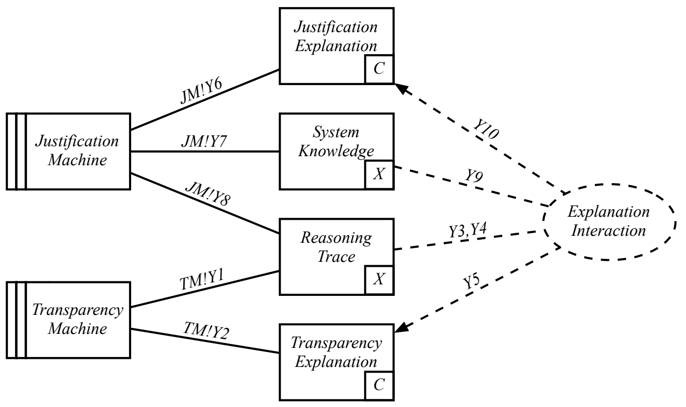

Fig. 9. Transparency and Justification Explanation

We want to integrate the explanation sub problems described by the two problem frame diagrams for the *Transparency* and the *Justification* goal to model the explanatory capabilities of the system. This combination is covered in the frame depicted in Figure 9. This model is a decomposition of the lower part of the *Interactive Ambient Intelligence Frame* from Figure 3. In the work presented here, we decide upon which of the two explanations to present as a function of the user's level of competence. That is, expert users are subject to transparency explanations and novice users to justification explanations [22].

By integrating Figure 8 modelling the ambient intelligence capabilities without explicit user interaction and Figure 9 capturing direct user involvement exemplified for explanations, we have a model (albeit simplified) of our explanation-aware ambient intelligent system. We can now re-visit the example problem described above. The expert user physician wishes to know how the combination of information displayed was reached. According to the transparency explanation problem frame, this can be achieved by displaying the reasoning trace. This can for example be done by showing that the top task 'Pre-ward round' was selected as a function of the classification, by displaying how the decomposition tree looks like, and by supplying information about the agents selected.

For the justification explanation, the novice user physician would like to know why this combination of information is any good. This can be achieved by relating the reasoning trace to the domain model of the system. For example, according to the domain model, the 'Acquire Medication' task could be satisfied not only by the *Patient Chart* but also by the *Electronic Patient Record.* However, as the *Electronic Patient Record* agent was busy serving other requests only the *Patient Chart* could respond to this request.

6.2 Analysing the Existing Application

The results of our ethnographical study are pointing towards the necessity to support four of the five different user goals introduced by Sørmo et al. [2], namely *transparency*, *justification*, *relevance*, and *conceptualisation*. This can be expressed in design specification documents which explicitly include the explanatory needs. When we look at the existing application, we can see that it does support the *transparency*, *conceptualisation*, and *justification* goals, where the latter is even only supported partially.

The fact that the system lacks certain explanatory capabilities is hardly surprising since they were not the main focus of the earlier implementation. However, the use of problem frames in general and explanation problem frames in particular helps us in identifying the deficiencies of the existing design, understanding and communicating explanatory needs, as well as exploring possible solutions to overcome these deficiencies.

Since we have started with data from an existing workplace analysis, we have not tested the hypothesis that (explanation) problem frames can be put to use in communicating with prospective users during the requirements elicitation phase. But we assume problems frames can enhance the cooperation between the requirements engineers and these users, as indicated by Phalp and Cox [13].

In the requirements analysis, introducing explanation frames facilitates the explication and formalisation of the findings of our ethnographical study and thereby deepens our understanding of the problem domain.

The use of problem frames as a method during requirement specification aids us in checking the completeness of the specification and helps us to incorporate explanatory needs which could otherwise be overlooked. This should also lead to a design which encompasses more features. If we had done the original system specification with the help of (explanation) problem frames, the missing support for the *relevance* goal would have been uncovered.

An explanation aware requirements specification is also fruitful in the transition from design to implementation. Earlier work by Roth-Berghofer and others has coupled explanation goals with the knowledge containers of case-based reasoning systems [23]. Having an explicit representation of explanation goals helps in identifying requirements for the knowledge containers, easing the way from a specification document to the structure and content of the knowledge containers.

7 Conclusion and Future Work

We have suggested the use of problem frames for analysing and specifying requirements for explanation-aware ambient intelligent systems. By introducing explanation patterns we have enhanced the toolbox for designing ambient intelligent systems.

We have shown how the use of explanation specific problem frames can help in recognising and explicating design requirements resulting from the necessity of intelligent systems to be able to explain their own reasoning and behaviour.

There are several issues that we have not addressed in this paper and which are left for further work. First and foremost, we have to explore further the relation between the design documents and the actual implementation. Our results show that problem frames help us to identify which explanatory knowledge and mechanism should be provided, but the methods for the next step in identifying the missing "knowledge containers" and suggesting remedies have to be extended over the existing work on the relation between explanation goals, explanation kinds, and knowledge containers.

References

1. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: ISTAG scenarios for ambient intelligence in 2010. Technical report, IST Advisory Group (2001)
2. Sørmo, F., Cassens, J., Aamodt, A.: Explanation in case-based reasoning – perspectives and goals. Artificial Intelligence Review 24, 109–143 (2005)
3. Leake, D.: Goal-based explanation evaluation. In: Goal-Driven Learning, pp. 251–285. MIT Press, Cambridge (1995)
4. Jackson, M.: Problem Frames – Analysing and Structuring Software Development Problems. Addison-Wesley, Boston (2001)
5. Cassens, J., Kofod-Petersen, A.: Designing explanation aware systems: The quest for explanation patterns. In: Roth-Berghofer, T.R., Schulz, S., Leake, D. (eds.) Explanation-Aware Computing – Papers from the 2007 AAAI Workshop. Number WS-07-06 in Technical Report, pp. 20–27. AAAI Press, Vancouver (2007)
6. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language. Oxford University Press, New York (1977)
7. Fensel, D., Benjamins, R., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., Musen, M., Motta, E., Plaza, E., Schreiber, G., Studer, R., Wielinga, B.: The component model of upml in a nutshell. In: WWW Proceedings WICSA1, 1st Working IFIP Conference on Software Architectures, San Antonio, Texas (1999)
8. Bergmann, R., Althoff, K.D., Breen, S., Göker, M., Manago, M., Traphöner, R., Wess, S.: Developing Industrial Case-Based Reasoning Applications, 2nd edn. LNCS, vol. 1612. Springer, Berlin (2003)
9. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., de Velde, W.V., Wielinga, B.: Knowledge Engineering and Management – The Common KADS Methodology. MIT Press, Cambridge (2000)
10. Avgeriou, P., Zdun, U.: Architectural patterns revisited – a pattern language. In: Proceedings of the tenth European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, pp. 1–39 (2005)
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston (1995)
12. Hatebur, D., Heisel, M.: Problem frames and architectures for security problems. In: Winther, R., Gran, B.A., Dahll, G. (eds.) SAFECOMP 2005. LNCS, vol. 3688, pp. 390–404. Springer, Heidelberg (2005)
13. Phalp, K., Cox, K.: Picking the right problem fram – an empirical study. Empirical Software Engineering 5, 215–228 (2000)
14. Hall, J., Rapanotti, L.: Problem frames for sociotechnical systems. In: Mate, J.L., Silva, A. (eds.) Requirements Engineering for Sociotechnical Systems, pp. 318–339. Idea Group Publishing, USA (2005)
15. Kofod-Petersen, A., Cassens, J.: Explanations and Context in Ambient Intelligent Systems. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 303–316. Springer, Heidelberg (2007)

16. Kofod-Petersen, A., Mikalsen, M.: Context: Representation and reasoning – representing and reasoning about context in a mobile environment. Revue d'Intelligence Artificielle 19, 479–498 (2005)
17. Aamodt, A.: Knowledge-intensive case-based reasoning in CREEK. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 1–15. Springer, Heidelberg (2004)
18. Gundersen, O.E., Kofod-Petersen, A.: Multiagent based problem-solving in a mobile environment. In: Coward, E. (ed.) Norsk Informatikkonferance 2005, NIK 2005, pp. 7–18. Institutt for Informatikk Universitetet i Bergen (2005)
19. Cassens, J., Kofod-Petersen, A.: Using activity theory to model context awareness: a qualitative case study. In: Sutcliffe, G.C.J., Goebel, R.G. (eds.) Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference, pp. 619–624. AAAI Press, Melbourne (2006)
20. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: Jade – a white paper. TILAB EXP "in search of innovation" 3, 6–19 (2003)
21. Kofod-Petersen, A., Aamodt, A.: Contextualised Ambient Intelligence Through Case-Based Reasoning. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 211–225. Springer, Heidelberg (2006)
22. Mao, J.Y., Benbasat, I.: The use of explanations in knowledge-based systems: Cognitive perspectives and a process-tracing analysis. Journal of Managment Information Systems 17, 153–179 (2000)
23. Roth-Berghofer, T.R., Cassens, J.: Mapping Goals and Kinds of Explanations to the Knowledge Containers of Case-Based Reasoning Systems. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 451–464. Springer, Heidelberg (2005)