

Evaluation and Refinement of a Design Framework for Generating Dependable Virtual Companions for Later Life

Dennis MACIUSZEK, Johan ABERG and Nahid SHAHMEHRI
*Department of Computer and Information Science
Linköpings universitet, 581 83 Linköping, Sweden*

Abstract. Resources in care for frail older people are limited. Electronic assistive technology can help maintain and improve older people's quality of life, however high requirements need to be put on such systems. As part of an iterative design process, this paper evaluates a framework for the design of interactive, multifunctional, and personalised assistive technology – virtual companions – in relation to previously defined dependability requirements. By means of scenario-based testing of a prototype implementation, we refine the first version of the design framework. The paper concludes with a reflection on our method, and outlines the iteration steps to follow.

Keywords. Assistive technology, elderly, virtual companion, dependability, personalisation, iterative design, multi-agent system

Introduction

On Wednesday 9 February 2005, local newspapers in Östergötland, Sweden reported the case of a 90-year old man who had died in his flat connected to a nearby assisted living facility. None of the staff on duty had noticed any of the six alarms he had sent, yet there also seems to be no obvious fault in the technology. It is the third time in three years that a similar incident has happened in the same town. [1, 2] At the same time, the county administration publishes a report showing deficits in care for people with dementia, who at several institutions are locked in during the night since there is no staff available to protect them from unwanted visits by other residents. [3]

The above examples give an indication of today's shortage of resources to ensure frail older people's health and quality of life. Formal care for a person is always limited to a certain amount of hours [4, 5], while informal caregivers need to take care of their own lives as well. *Electronic assistive technology* (EAT) is used to help maintain or improve quality of life, yet it becomes clear from the tragic example above – no matter whether the error was human- or machine-made – that very high dependability requirements must be put on such technology. In addition to professional routines and trustworthy staff, designers of assistive technology need to make their contribution so that old people can depend on installed technologies.

Our own work concerns the development of EAT that integrates different assistive applications in one system, interconnecting them. The project shares many visions with

the idea of a *smart home* (e.g. [6, 7]), yet the emphasis is on introducing an interactive presence into the house, rather than on making existing objects smart. We have thus adopted for it the metaphor of a *virtual companion*. Examples of such systems would include personal service robots like the Nursebot [8] and smart phones that offer various assistive services [9].

A main project focus is on personalisation – due to the high inter-individual and intra-individual variance in frail older people's needs (dynamic diversity [10]). To obtain interactive, multifunctional EAT, we do not aspire a uniform design of virtual companions, but rather we proposed a *design framework* consisting of a *generic multi-agent architecture* and an *instantiation process* that populates it with special-purpose software agents to support individual needs. [11] Our long-term vision is to automate the instantiation process, and to give users, caregivers, and experts a toolkit that will automatically generate personalised companions based on their own selections.

We have previously identified 9 non-functional requirements that ought to be fulfilled to make virtual companions and related aids (like a smart home) *dependable*. [11] The requirements are:

Adaptability: It must be easy to add, update, and remove applications and data in order to suit the individual elderly user. (cf. [12, 13])

Availability: The system must hold applications ready, and if necessary help proactively. (cf. [14])

Relevance: The system must support a variety of everyday life situations in appropriate ways. (cf. [15])

Acceptability: The system must be useful and easy to use. (cf. [16])

Reliability: The system must anticipate, notice, and take care of errors. (cf. [14])

Competence: The system must provide good service by effectively processing knowledge obtained from user, caregivers, and experts, and by taking into account current environment context.¹ (cf. [17])

Safety: The system must not harm user or environment.² (cf. [13, 14])

Adaptivity: During usage, the system must adjust its behaviour to user goals, user competence, and user interests. (cf. [12])

Security: The system must restrict access to applications and personal data to authorised persons. (cf. [14])

1. Objectives

In [11] we argued why our proposed design framework for the creation of virtual companions – in theory – is suited to produce assistive software that elders can depend on. In this paper, we investigate how this method works in practice.

In order to get closer to the vision of the virtual companion and the companion-generating toolkit – both systems of high complexity –, we employ an iterative design method, shown in Figure 1.

First, we collected the dependability requirements (see **Introduction**). Then we defined a design framework with the prospect of fulfilling the requirements. It is described in Section 2. This paper contributes a technical evaluation of this framework.

¹ 'Competence' refers to a requirement, whereas 'user competence' refers to capabilities of an elderly user.

² The term 'safety' is used for both a requirement and an application area.

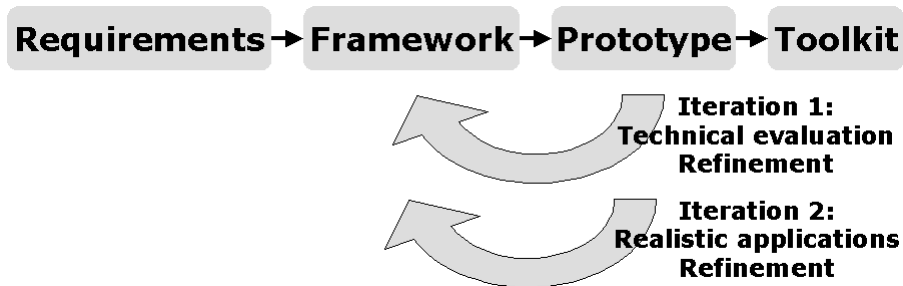


Figure 1. Iterative design

We implemented a prototype of the main framework mechanisms in Java, and ran a set of scenario-based tests on it, evaluating its compliance with the requirements. Section 3 describes the method, while Section 4 presents major results. The aim of this study was to arrive at a set of refinements to be fed back into the framework design, thus creating a second version of the framework. It will be evaluated at least once more – that time with realistic applications and the direct involvement of users in real environments. Subsequent refinement will result in a third framework, and then a third prototype, which will also yield the first modules of the envisioned companion-generating toolkit. In addition to discussing the results of Iteration 1, Section 5 returns to this methodology, and summarises our experiences so far. Section 6 concludes the paper.

2. Design Framework

Figure 2 illustrates the generic multi-agent architecture. Agents are regarded as autonomous software modules capable of goal-directed behaviour and communication with each other. An agent can act in a proactive way, i.e. it can notice when its goal is

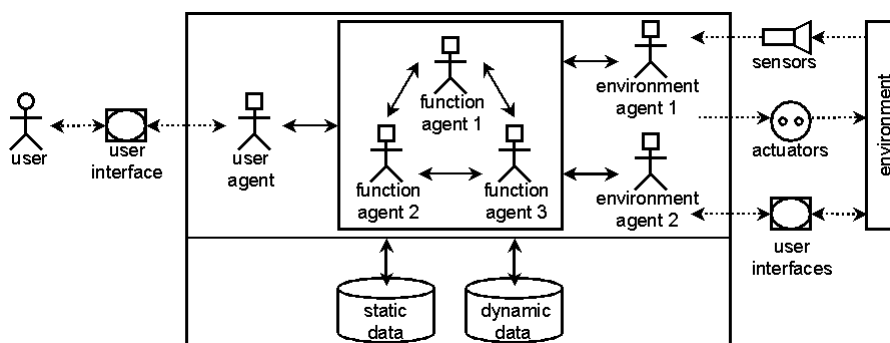


Figure 2. Generic multi-agent architecture

not fulfilled, and then it will do its best to attain that goal.

At the heart of the architecture, there are *function agents*. These realise a companion's applications. Each function agent is made up of one *function* component which is combined with one *application area* component. The function is a software component that provides the agent's algorithm. It can be seen as a reusable implementation of an EAT design pattern. Since our function agents are interactive agents, we based their behaviour algorithms on inspiration from real patterns of assistive interaction (cf. [18, 19]) between a caregiver and an older person. Our current design includes 15 such functions: *monitor*, *mediator*, *operator*, *reminder*, *rememberer*, *guide*, *trainer*, *informer*, *supplier of activities*, *speaking/writing aid*, *hearing/reading aid*, *listener*, *communicator*, *locator*, and *recommender*. These patterns were compiled from analysing literature on communication in care, among others [20, 21, 22] (with additional inspiration from [23]). They are currently being verified in a field study.

An application area can currently be any of 18 (hierarchically-structured) aspects of everyday life. These range from different activities of daily living (ADLs [24], IADLs [25]) to more general needs like *safety* or *social contact*. It contributes a goal (e.g. to cook a recipe, to be safe) and data to the function agent. This data is often partly dynamic, e.g. a process model of a certain procedural activity, and partly static, e.g. descriptions and explanations. Whereas dynamic data always belongs to a certain agent, static data is shared.

Function agents can communicate with each other in order to achieve more complex behaviour. For instance, a *safety monitor* notices that the sink basin is full, and delegates this problem to a *safety operator*, which turns off the sink tap, and opens the drain. Function agents can further communicate with each other in order to watch over each other. For instance, the *safety operator* mistakenly opens the window when it is cold. The *safety monitor* notices the effect of this – it is getting even colder –, and it warns the user about this.

In addition, function agents communicate with a *user agent* – a representation of the user in the system – in order to adapt behaviour, e.g. to individual user competence (static data, e.g. is the user physically capable of opening/closing the window?). They communicate with *environment agents* – representations of objects and persons – in order to be aware of current context (dynamic data updated through sensors) and to manipulate the environment (use an actuator, contact a human assistant). Consider the above cases of sensing water level or temperature, and operating tap, drain, or window. Sensors, actuators, and user interfaces – which are handled by user and environment agents – are currently regarded as external infrastructure.

During the instantiation process, the architecture is populated with agents chosen for a particular user, and filled with personal data.

3. Method

A prototype of the virtual companion framework was implemented, and evaluated according to the requirements. The emphasis was on having as many conceptual features of the design in the implementation as possible, even if that meant mixing advanced approaches with ad-hoc solutions. On the other hand, we did not aspire to implement every function and application area. Instead, we made the implementation extendable by keeping concepts like a goal or a process generic, by making use of

object-orientation, and by making even parts of agent behaviour reusable. For instance, the method to solve a problem is used by three function agent classes.

Implemented software components are the *monitor*, *reminder*, *guide*, *operator*, *communicator*, and *trainer* functions with the application areas *IADL* (instrumental activity of daily living [25]) and *safety*. For the sake of the prototype, we regarded an IADL as a concurrent process, and chose the preparation of food recipes as an example (for clarity of what the agents do, this application area will from now on be called *food preparation*). For safety, we implemented the need for a safe environment. The implementation contains two types of environment agents, *sensor worlds* (that model and manipulate current states of the environment via sensors and actuators) and *persons* (who can be contacted for help).

The architecture was instantiated with agents for a particular user, Mr Taylor. He has a mild dementia, but could remain independent during the day if helped with preparing lunch and the prevention of accidents. We restricted the supported environment to the kitchen where Mr Taylor has a cooker, a sink, and a window and heating for regulating temperature. The appliances are equipped with different sensors and actuators. Due to our intention of evaluating core technical aspects before realistic applications, Mr Taylor is a fictitious user, and the kitchen with sensors and actuators was simulated. Agents that were created by combining architecture components include a *food preparation trainer*, *food preparation guide*, *safety trainer*, *safety reminder*, *safety monitor*, *safety guide*, *safety operator*, *safety communicator*, three *sensor worlds*, two *persons* (Mr Taylor's son and the fire brigade), as well as the *user agent* as a representation of Mr Taylor. A detailed report on this case study was presented as a poster in [26]. The case of Mr Taylor was inspired by [27], yet the walking disability was exchanged by a dementia.

In order to evaluate framework design and prototype with respect to real-world requirements, we chose a method of scenario-based testing. Scenarios were extracted from personal interviews conducted in parallel to the implementation with 8 elders at a retirement home and an assisted living facility (mean age 85 years) and 6 caregivers at the same retirement home. Additional data came from a focus group interview we did with 4 caregivers of different institutions. One group of scenarios concerns incidents from real life, and a second group describes hypothetical situations that were described in response to a presentation of our design of Mr Taylor's virtual companion.

Extracted scenarios were filtered so that the chosen ones would guarantee a good coverage of the dependability requirements. From these, we generated technical test cases, performed the tests, documented the results, and then critically analysed them to yield framework refinements.

We were not able to achieve a good coverage for acceptability without considering real users and more sophisticated user interfaces. Like the UI, securing a network connection to the outside is regarded as external to the framework, and so far no special security mechanisms have been designed or implemented. Therefore, although we did generate a number of scenarios and tests for these requirements, we did in the end not perform them. Results would have been of limited significance.

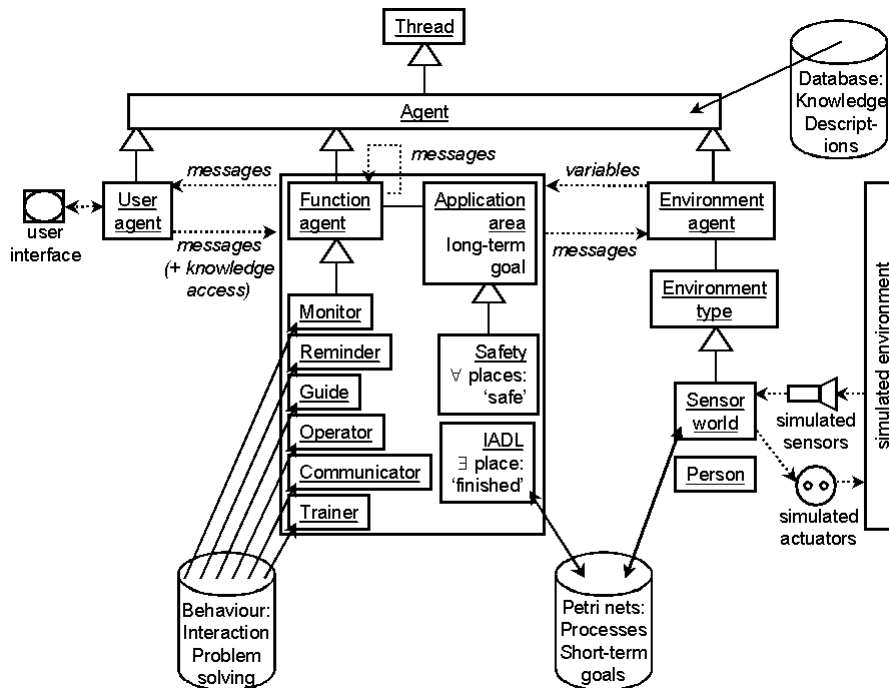


Figure 3. Prototype implementation

4. Results

4.1. Prototype Implementation

The prototype of the generic architecture was implemented in Java. Figure 3 depicts implemented Java classes as boxes, with white arrows where inheritance was used.

Each agent is a Java thread, i.e. a process which runs concurrently with other processes. Agent behaviour was implemented as infinite loops waiting for a reason to act. Such a reason can be the realisation that an agent's goal is not fulfilled, e.g. a *safety monitor*, which continuously checks environment processes – through variables shared with environment agents – notices that a cooker plate is too hot. It can then send a message to a *safety guide*, *operator*, or *communicator* in order to delegate the problem. Then the respective agent would itself have a reason to act, while the *safety monitor* concurrently continues monitoring. Agent behaviour may further be invoked by the user agent, which handles all interaction with the user through a user interface. In the implementation, these are dialogue windows in which the user answers simple questions (e.g. shown on touch-screens throughout the home). Notice that the current GUI was included for purposes of technical testing, and no attempt was made to create a comprehensive interface for real users. At times when no system-initiated interaction is going on, the user may request an application, e.g. guidance in food preparation,

which invokes a *food preparation guide*. Notice that the function agent may also become active on its own, namely when it offers to be of assistance at the user's preferred mealtime.

Agent behaviour algorithms can be rather straightforward, e.g. when a *safety reminder* tries to fulfil a short-term goal, and reminds the user of a certain task to do in a certain situation. Then it just follows a specified rule in the dynamic data. Alternatively, a rather complex problem may need to be solved, e.g. when a *safety operator* tries to attain its long-term goal (e.g. safety in relation to room temperature) by requesting the activation of actuators. To find out which actuators will correct the problem (e.g. it is cold), it will first reason on the basis of current context (the window is open, the heating is off) and a dynamic model of the respective *sensor world environment agent*. Dynamic data has been modelled with Petri nets, which are basically concurrent automata or transition systems. Problem solving is a search for all paths in Petri net reachability trees (cf. [28]), i.e. paths including the transitions (the tasks to do) from a configuration in which the agent's goal is invalid to a configuration in which it is valid. In choosing the best solution, complementary static data is considered (e.g. is the task of closing the window or turning on the heating automated?). For a *guide* or a *trainer*, this involves asking the user agent about the user's competence to do certain tasks, which is also knowledge stored in the static database.

For handling Petri nets, we chose the publicly available *Petri Net Kernel* library [29] and added own analysis algorithms. For static data, we created a *MySQL* database, which is accessed via Java's database interface *JDBC*. Except for monitoring processes where we employed shared variables, agents communicate with each other through a message passing mechanism realised with Java *pipes*. Exchanged messages have the syntax of *FIPA XML* [30]. This is used to facilitate the identification of an incoming message and its content. Yet, our agent collaboration does not follow the advanced FIPA semantics. Instead it was considered sufficient for the purpose of the prototype if just a few straightforward collaboration patterns like the delegation of a problem were implemented. XML is processed in our system using the *JDOM* API [31].

Once all generic components were implemented and tested, all that was left to do was the instantiation for Mr Taylor's personal companion. This involved

- creating the 14 particular agents (5 environment, 1 user, and 8 function agents) in a main companion class,
- adding 4 dynamic data processes for cooker, sink, and temperature behaviour, as well as an example recipe (using Petri Net Kernel's graphical editor),
- filling the database with static data, such as Mr Taylor's user competence and the user competence that tasks (transitions in the Petri nets) require, the expertise of human assistants (*person environment agents*) and the expertise required to handle problems (which are sets of Petri net places/states), as well as textual and graphical descriptions and explanations of problems and tasks.

Since the prototype was not installed in a real home, a graphically animated simulation of Mr Taylor's kitchen – including simulated sensors and actuators – was created to provide an environment for subsequent evaluation.

4.2. Evaluation of Prototype and Design

We turn now to the results of the scenario-based tests. A selection has been made so that those results are summarised and discussed that cover their respective requirements well, and at the same time led to significant information for the goal of refining our framework. Notice that no tests were carried out for acceptability and security.

4.2.1. Adaptability

It was already obvious from interviewing 8 older people that every potential user will be a unique individual with unique, individual needs. We heard the complaints of a woman who moved into an assisted living facility only to find that mostly leisure activities were offered when she would rather have needed safety monitoring, shopping possibilities, and help to get to the hairdresser. Yet, 'it is not easy to find the right things for 71 residents', she said. However, this is our main concern in the vision of a companion-generating toolkit. Is a toolkit for composing very personalised assistive technology feasible?

Our case study with the fictitious Mr Taylor indicates that reuse and recombination of generic software components can be an efficient tool in overcoming the variety-of-needs bottleneck. Mr Taylor needs 8 assistive technology applications. Traditional design would give him 8 different devices for these needs. Each of them would require designing and implementing one set of data and one algorithm that works on the data from scratch. In our terms, these would be $8 \times 2 = 16$ components: *food preparation + trainer + food preparation + guide + safety + trainer + safety + reminder + safety + monitor + safety + guide + safety + operator + safety + communicator*. In our design with software components, on the other hand, we were able to reuse the *food preparation* and *safety* components (data) and the *trainer* and *guide* components (algorithms). Thus, we needed only create $2 + 6 = 8$ components: *food preparation + trainer + guide + safety + reminder + monitor + operator + communicator*.

In addition, it was pointed out by caregivers that especially dementia is a disease that develops quickly. Today's aid may be useless tomorrow. We evaluated this by assuming Mr Taylor would no longer understand a *guide* or a *trainer*, but instead would need an additional *food preparation communicator* to summon human help also when encountering a food-related difficulty. Doing this update required easy modification of 6 lines in the code. Since the added function agent worked perfectly with existing data, we considered that test a success.

Things got more difficult when we added a second recipe to evaluate the scenario described to us when Mr Taylor's son would sit down with his father and plan the week's meals. Both static and dynamic data had to be updated, with reuse possible only when specifying required user competence, but not in defining actual steps in a recipe. That said, the modifications were quite systematic, and could be facilitated a lot in a semi-automatic companion creation toolkit. Entered data was used by the existing agents without problems.

4.2.2. Availability

According to the interviews, frail older people often do not actively request help. An alarm might not be reachable in the case of an accident, people might forget to request help, they might not want to bother caregivers, not notice when e.g. water is running, etc. Here, our proactive agents that warn when the basin is full or offer food

preparation assistance at the user's preferred meal times can be of good help. Like human caregivers who often have to assist in different ways at the same time, being helped by one agent does not block another agent's concurrent behaviour. If a *safety monitor* needs to deliver a warning or a *safety reminder* needs to remind during a *food preparation guide's* explanations, they do so. However, the tests showed us that in our framework the sequence of such interleaved behaviour is arbitrary, and in fact the thread scheduling often prioritised a *guide* over a *reminder*, which from the user's point of view was questionable.

4.2.3. Relevance

Two interviewees found monitoring the actual *food preparation* activity more relevant in everyday life than a *safety monitor* for the cooker. In cooking, food may boil over or scorch. In 4.2.1, we stated that it is technically easy to create new applications by just recombining available components. But does this result in relevant support? The idea with a *food preparation monitor* would be that a user has left the cooking area, and needs to be reminded that a cooking process is going on. We realised this function agent by adding two lines of code, and found its collaboration with the *food preparation guide* working technically correctly. However, far too many warnings were generated. This was because the goal of the respective application area was not formulated precisely enough. In addition, it was not considered whether a current cooking situation actually involved boiling or frying something, and whether the user actually was in the cooking area.

4.2.4. Acceptability

Some assistance in care has an unclear purpose: Do memory training sessions train memory, or are they recreational or social activities? 'Seated dance' may be dancing or physical exercise. Shopping at the market may be shopping or an excursion. An old magnifying glass may be used for reading or just have nostalgic value. We defined a number of tests that would have indicated whether our virtual companion would be useful to attain its goals, or whether interacting with it would have been perceived as play. Yet, such laboratory tests would probably not have predicted real-life usage and usefulness.

We received comments from interviewees saying that often technology in general, and our design in particular, were too cognitively demanding for elders, especially those with dementia. From the comments, we generated tests that would have checked whether descriptions appear in a logical sequence, whether turning off the cooker looks and reads the same for all applications, how many interaction steps would be needed, and what removing the more demanding *guides* and *trainers* would have meant for acceptability. However, to comprehensively assess this requirement, our prototype and its GUI would have needed to be used and accepted by real users – something for which its technically-oriented design and implementation were not good enough yet.

4.2.5. Reliability

This is the case of the tragic newspaper story. Assistive technology was available, but it did not save the man's life because of some error – either in the machine or in a human assistant who did not respond. If a *safety communicator* malfunctions in our design and implementation, alternative function agents with the same goal may have been installed

that will take care of a problem – given that the user will e.g. understand a *guide*, or that there is an actuator that the *operator* can request to activate in order to solve the situation. This is because the *safety monitor* continues monitoring, and notices the same problem again. In addition, our *communicators* send a signal to an assistant every hour. In the current prototype, this is implemented as an e-mail, and in the test it was easy to notice that something was wrong when the usual mail did not appear in the inbox.

That said, we did compromise reliability with some design decisions – things that were not obvious to us in the first place. Like one interviewed caregiver, we were thinking the companion would be very **competent** if it reminded the user to turn off the cooker after finishing a cooking activity only when the cooker was actually on (context-awareness). That however made the *safety reminder* fail right away when we damaged the simulated sensor in the cooker that normally reports that the cooker is on.

Another decision was not to offer a *guide* but only an *operator* and a *communicator* in a situation of medium urgency, since guiding might take too long, thus compromising **safety**. We decided to offer neither *guide* nor *operator* but only a *communicator* in a situation of high urgency, since both might take too long or not work, again compromising **safety**. In the case of a fire, the fire brigade should be called right away. However, having a fire and a malfunctioning *communicator*, the user was left with no more working options.

In addition to redundant function agents, i.e. agents serving the same goal that can replace each other, reliability is supported by concurrency in process models, which leads to an increase in potential problem solutions. We tested the scenario where the microwave was broken and a *food preparation guide* needs to describe recipes referring to the old-fashioned cooker instead. The test succeeded, as indeed the other solution path was chosen.

The difficulty here was rather damaging the microwave for the test. First of all, we did not include an environment agent representing a microwave. But even if we had, there is no connection between a recipe process and the actual model of a used appliance. What we did instead was set the user's competence in using a microwave oven to false.

4.2.6. Competence and Safety

When presented with the design of the virtual companion for Mr Taylor, several of the interviewed caregivers were sceptical that complex activities such as food preparation could be understood by a computer. ‘You would need to develop a program that includes all potential options.’ ‘A lot of knowledge is necessary. Everything must be in it: Potatoes, vegetables, ... One would also need to describe where all the items are.’ ‘Cooking is a complex activity: deciding for a recipe, buying the corresponding ingredients, etc.’

With our technical prototype we could not test such realistic scenarios. We saw however that it is reasonably easy – just a bit cumbersome – to manually add a new recipe taken from a Web site. In the envisioned companion creation toolkit, input from the user, caregivers, and experts would be facilitated more by means of semi-automatic knowledge acquisition. Occupational therapists build advanced models of food preparation in order to assess user competence in relation to these [32, 33]. Pigot et al. [34, 35] incorporated such models into a cognitive assistance system. Our design is capable of storing and using realistic models, the only problem being that much of the information entered is syntactical (a Petri net graph) rather than semantic. This is easier

to process (abstract path search in a reachability tree), yet it may no longer work with more advanced agent collaboration, which would be needed to realise the scenario description above.

Some semantic information in our design was already tested successfully, namely user competence (see 4.2.7) and the urgency of a situation. Elders and caregivers would very much like to complement alarms with urgency information, and participants in the focus group interview wanted an urgent situation to be supported by an *operator* rather than *guide*-like functionality. Both was found to work in our implementation.

It was mentioned earlier that denying the user a *guide* may compromise **reliability**, and that the same was found for a context-awareness mechanism that allows a cooker reminder only when the cooker is on. Otherwise, the tests we ran on context-awareness – for both competence and safety – were successful, with the exception that time is not considered enough. With basic Petri nets, it cannot be modelled how long something needs to cook (no real-time tasks), or how long the physical process of filling a sink basin with water takes (discrete states instead of continuous time).

It remains to be studied how electronic assistance can be so competent that it can realistically enable a frail older person to carry on with activities. When such empowerment succeeds though, it is important that it does not negatively affect safety. It was a recurring theme in the interviews that people tend to give up activities such as food preparation or walking not because of a lack of user competence, but because they did not feel safe in doing them anymore.

4.2.7. Adaptivity

A user goal is a general need that corresponds directly with the goal in an application area component. In a real world scenario described by the focus group participants, an elderly lady refused all attempts to explain to her how her remote control for the TV works. She would rather get up and adjust the TV manually. This illustrates that even if one has certain technology one may opt not to use it. The lady may however change her mind should she ever become bed-ridden. That is to say, applications should be automatically activated and deactivated even during usage. Although we have only sketched (and not implemented) this feature so far, we mention it here since it illustrates another potential trade-off. If we include deactivation, it might harm **availability** and **reliability**.

Adaptivity to the user's competence in doing certain tasks was tested successfully. When a *guide*'s problem solving method chooses one of several candidate solutions, it prefers those which the user is more likely to handle. On the contrary, a *trainer* practises tasks that are difficult. Although this does work for simple applications, our user modelling, which was inspired by occupational therapy and mathematical psychology models [32, 33, 36], still does not model e.g. a 30% user competence or the physical but not cognitive user competence to do a task. In addition, we only sketched learning of changed user competence so far.

Adaptivity to the user's interests has been considered in specifying preferred meal times and a preferred bedtime. In a care institution, residents need to adapt to certain institutional schedules themselves, e.g. meal times which they are not used to. This can create unnecessary confusion. In our tests, the *food preparation guide* correctly offered its services at the times the user prefers, and he was also reminded to close the window at his usual time of going to bed. User habits can of course change just as user competence, but learning was not implemented yet.

4.2.8. Security

We generated two scenario-based tests for security: (1) Try to intrude into a *shopping operator* application, and manipulate it so that the user orders unwanted things. (2) As an unauthorised user, try to achieve that Mr Taylor will be guided to eat each day the food that you want him to eat. Since we did not carry out these tests, we refer instead to [37] for a security analysis of certain home automation applications.

4.3. Refinement

We present here a selection of some major improvements of the virtual companion design framework after the first iteration.

The new companion design framework makes comprehensive use of *semantic information*. Previously, semantic concepts like different levels of *risk* were used in the *safety* application area component. From these, *urgency* was computed. This is also needed for other application areas like IADLs in order to make these **relevant** and **competent** (and actually also more **acceptable**). We have already begun experimenting with semantic XML models of recipes that are semi-automatically extracted from Web pages. If these could be related to a recipe ontology, then the adding of new recipes (**adaptability**) would also be facilitated.

On the way to realising realistic (**relevant** and **competent**) applications, *agent collaboration* must be refined. In order to be able to combine function agents for complex help behaviour, there must be more than delegation of syntactical problems and indirect monitoring of agents' effects on process variables. A *shopping writing aid* will send shopping lists to *shopping operators* which order ingredients from which a *food preparation recommender* compiles recipes. Currently, functions represent single-agent design patterns realised as singular software components. Now there will be multi-agent design patterns and formal connections of function components. One such connection is the attachment of a *safety monitor* to any *guide* in order to ensure that empowerment of a person does not result in a loss of **safety**. To attain **reliability**, *safety communicators* will be complemented by *guides* and *operators*. Maybe the 90-year-old's life could have been saved by a *guide* giving the emergency number, or by an *operator* opening the door so he could have called for help?

Another benefit of well-defined collaboration patterns and component interfaces lies in the future where we can imagine different companies supplying their own companion components, which must allow for being inter-combined. In our interviews, we described a fall detection system – a new *safety monitor* – to a physically weak elderly lady. Her immediate comment was that it would be very useful, but she would have to be able to connect it to her current alarm device – a *safety communicator* application acquired from and driven by a commercial care service.

User model knowledge becomes more refined with fuzzy values and prerequisite relationships. In order to be able to 'turn off' something, one must be able to physically reach and to cognitively understand how it works. Changing user competence and user interests are learnt through interpretation of the user's actions. This will realise real **adaptivity**. In addition, the performance of agents in doing certain tasks is learnt in order to improve **reliability**.

Competence, **safety**, and **relevance** are supported by more *context-awareness*. The user agent gets access to sensors that indicate the user's position in the environment. Processes model *time*. They will also be *interconnected* (e.g. recipes with

the cooker). We are currently experimenting with the component-based modelling language Modelica [38], which can also explicitly describe and simulate real-time and continuous-time behaviour.

There will be a systematic definition of *application area goals*. We learnt from the **relevance** tests that plugging together components is not always straightforward. In addition, an *urgency* function will be defined for every application area, so that the user agent can prioritise interaction with the function agent that has the most urgent concern (**availability**).

5. Discussion

The above refinements must be evaluated in a second iteration, that time with users and realistic applications. The second evaluation should further evaluate the different cases of trade-offs that were discovered. For instance, participants in the focus group preferred only having full automation (i.e. an *operator*) over including potentially confusing interaction such as *guiding* – at least for certain users. In the evaluation however, we saw that suppressing or not including certain functions might reduce **reliability**. When Dewsbury et al. [39] discuss the dependability of a smart home simulation toolkit, they conclude that extensive **adaptability** may reduce **reliability** and **safety**. Are these trade-offs natural, or can they be prevented?

Looking back at our method, we realise how important it was to begin with a small-scale evaluation of core technical features. Very logical design decisions can yield unexpected results, and very basic design alternatives become visible that in large-scale real-life tests are easily concealed by other factors.

6. Conclusions and Future Work

We have given insight into the first iteration of a design process that has the goal of creating dependable virtual companions for later life. We evaluated our design framework according to non-functional requirements, and outlined selected features of its resulting second version. We made use of an evaluation method we believe is relevant for any design of multifunctional EAT.

We have addressed many technical questions and presented a potentially robust design framework. Now the question is: How does it scale to the variety of needs of real users? What are the relevant applications for older people's quality of life? Can our method of reuse create these from a feasible set of software components? Given these components have been implemented, can the process of generating personalised virtual companions be automated?

Our next step is the identification of relevant and useful applications by means of a field study. We are investigating what function and application area components are needed, and how and to what extent they can be combined. Our next iteration step is then to design and implement realistic applications for real users with the help of the second design framework and a second prototype. These virtual companions will be evaluated in experiments, and we will gain insight into both the way towards the envisioned toolkit and the scalability of the companion idea to a variety of people and the complexity of everyday life.

Acknowledgements

This research is done in collaboration with the National Institute for the Study of Ageing and Later Life at Linköpings universitet. It is supported in part by the Swedish Council for Working Life and Social Research. We owe special thanks to Gerrit Dolle, Brigitte Wauer, and Hildegard Maciuszek for arranging the interviews, as well as to all participants.

References

- [1] Östgöta Correspondenten, 9 February 2005.
- [2] Extra Östergötland, 9 February 2005.
- [3] Länsstyrelsen Östergötland. Natten är också en del av dygnet.... [The night is also a part of a day....] Rapport: 2005:3, Sociala enheten 2005-02-09. In Swedish.
- [4] K. Liu, K. G. Manton, and C. Aragon. Changes in home care use by disabled elderly persons: 1982–1994. *The Journals of Gerontology Series B: Psychological and Social Sciences*, 55(4):S245–S253, 2000.
- [5] The National Board of Health and Welfare. Statistics – Social welfare. Care and services to elderly persons 2002. Official statistics of Sweden, 2003.
- [6] C. D. Kidd, R. J. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner, and W. Newstetter. The Aware Home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings – CoBuild '99*, Pittsburgh, PA, USA, 1999.
- [7] R. Orpwood. Dependability issues in smart house design. In *Proceedings of the First 'HEAT: The Home and Electronic Assistive Technology Workshop'*, pages 13–17, York, UK, 2004.
- [8] G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Magaritis, M. Montemerlo, J. Pineau, N. Roy, J. Schulte, and S. Thrun. Towards personal service robots for the elderly. In *Workshop on Interactive Robotics and Entertainment*, Pittsburgh, PA, USA, 2000.
- [9] W. Mann and A. Helal. Smart phones for the elders: Boosting the intelligence of smart homes. In *AAAI 2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care*, pages 74–79, Edmonton, Alberta, Canada, 2002.
- [10] P. Gregor, A. F. Newell, and M. Zajicek. Designing for dynamic diversity – interfaces for older people. *Proceedings of the Fifth International ACM Conference on Assistive Technologies (ASSETS 2002)*, pages 151–156, Edinburgh, UK, 2002.
- [11] D. Maciuszek, N. Shahmehri, and J. Aberg. Dependability requirements to aid the design of virtual companions for later life. In *Proceedings of the First 'HEAT: The Home and Electronic Assistive Technology Workshop'*, pages 51–60, York, UK, 2004.
- [12] G. Dewsbury, I. Sommerville, K. Clarke, and M. Rouncefield. A dependability model for domestic systems. In S. Anderson, M. Felici, and B. Littlewood, editors, *Computer Safety, Reliability, and Security: 22nd International Conference, SAFECOMP 2003, Proceedings*, number 2788 in Lecture Notes in Computer Science, pages 103–115, Edinburgh, UK, 2003. Springer-Verlag.
- [13] C. A. Miller, K. Haigh, and W. Dewing. First, cause no harm: Issues in building safe, reliable and trustworthy elder care systems. In *AAAI 2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care*, pages 80–84, Edmonton, Alberta, Canada, 2002.
- [14] A. Avizienis, J.-C. Laprie, and B. Randell. Fundamental concepts of dependability. UCLA CSD Report 010028, Computer Science Department, University of California, Los Angeles, USA, 2001.
- [15] P. Carlshamre. *A Usability Perspective on Requirements Engineering*. Dissertation, Linköpings universitet, Sweden, December 2001.
- [16] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):318–340, 1989.
- [17] B. M. Muir. Trust in automation: Part I. Theoretical issues in the study of trust and human intervention in auto-mated systems. *Ergonomics*, 37(11):1905–1922, 1994.
- [18] I. Sommerville, D. Martin, and M. Rouncefield. Informing the requirements process with patterns of cooperative interaction. *International Arab Journal of Information Technology*, 1(1), 2003.
- [19] A. Crabtree. *Designing Collaborative Systems. A Practical Guide to Ethnography*. Springer-Verlag, London, UK, 2003.

- [20] E. Cedersund and C. Nilholm. *Samtal I äldreomsorgen. Samspelet mellan omsorgspersonal och äldre med Alzheimers sjukdom*. [Conversation in care for the elderly. The interplay between care staff and elderly people with Alzheimer's disease.] Studentlitteratur, Lund, Sweden, 2000. In Swedish.
- [21] H. Hazan. *Managing Care in Old Age. The Control of Meaning in an Institutional Setting*. State University of New York Press, Albany, NY, USA, 1992.
- [22] W. A. Rogers, B. Meyer, N. Walker, and A. D. Fisk. Functional limitations to daily living tasks in the aged: A focus group analysis. *Human Factors*, 40(1):111–125, 1998.
- [23] M. Fleck, M. Frid, T. Kindberg, E. O'Brian-Strain, R. Rajani, and M. Spasojevic. From informing to remembering: Ubiquitous systems in interactive museums. *Pervasive Computing*, 1(2):13–21, 2002.
- [24] S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson, and M. W. Jaffe. Studies of illness in the aged. The index of activities of daily living. A standardized measure of biological and psychological function. *Journal of the American Medical Association*, 185:914–919, 1963.
- [25] M. P. Lawton and E. M. Brody. Assessment of older people: Self-maintaining and instrumental activities of daily living. *Gerontologist*, 9:179–185, 1969.
- [26] D. Maciuszek, N. Shahmehri, and J. Aberg. User involvement in the design of software to assist people with dementia: Creating a personalised virtual kitchen companion. In *ENABLE Final Conference. Challenges in Dementia Care: Can Technology Help People with Dementia?*, Oslo, Norway, 2004.
- [27] C. Glogoski and D. Foti. Special needs of the older adult. In L. W. Pedretti and M. B. Early, editors, *Occupational Therapy: Practice Skills for Physical Dysfunction*, chapter 51, pages 991–1012. Mosby, St Louis, MO, USA, fifth edition, 2001.
- [28] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.
- [29] <http://www.informatik.hu-berlin.de/top/pnk/>
- [30] FIPA ACL Message Representation in XML Specification. FIPA standard SC00071E, 2002.
- [31] <http://www.jdom.org/>
- [32] N. Josman and S. Birnboim. Measuring kitchen performance: What assessment should we choose? In *Scandinavian Journal of Occupational Therapy*, 8:193–202., 2001.
- [33] R. E. Levine and C. R. Brayley. Occupation as a therapeutic medium. A contextual approach to performance intervention. In C. H. Christiansen and C. M. Baum, editors, *Occupational Therapy. Overcoming Human Performance Deficits*, chapter 22, pages 590631. Slack, Thorofare, NJ, USA, first edition, 1991.
- [34] H. Pigot, B. Lefebvre, J.-G. Meunier, B. Kerhervé, A. Mayers, and S. Giroux. The role of intelligent habitats in upholding elders in residence. In *5th International Conference on Simulations in Biomedicine*, pages 497–506, Slovenia, 2003.
- [35] H. Pigot, A. Mayers, S. Giroux. The intelligent habitat and everyday life activity support. In *5th International Conference on Simulations in Biomedicine*, pages 507–516, Slovenia, 2003.
- [36] D. Albert and T. Held. Component based knowledge spaces in problem solving and inductive reasoning. In D. Albert and J. Lukas, editors, *Knowledge Spaces: Theories, Empirical Research, Applications*, pages 15–40. Lawrence Erlbaum Associates, Mahwah, NJ, 1999.
- [37] A. Herzog and N. Shahmehri. Towards secure e-services: Risk analysis of a home automation service. In *Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec)*, pages 18–26, Copenhagen, Denmark, 2001.
- [38] P. Fritzon. *Principles of object-oriented modeling and simulation with Modelica 2.1*. IEEE Press, Piscataway, NJ, USA, 2004.
- [39] G. A. Dewsbury, B. J. Taylor, and H. M. Edge. Designing dependable assistive technology for vulnerable people. *Health Informatics Journal*, 8(2):104–110, 2002.