HCM-L: Domain-Specific Modeling for Active and Assisted Living

Heinrich C. Mayr, Fadi Al Machot, Judith Michael, Gert Morak, Suneth Ranasinghe, Vladimir Shekhovtsov and Claudia Steinberger

Abstract Modeling and modeling methods are crucial for information systems engineering but are seldom seamlessly integrated into all phases of development and operation: Practitioners challenge the benefits of modeling and complain about the confusing variety of concepts with overlapping semantics, symbols and syntactic rules of today's standardized, "universal" modeling languages. Therefore, domain-specific modeling languages (DSMLs) are gaining increasing popularity: they are lean and convenient, support the productivity of modeling, and help to increase model quality and comprehensibility. There are, however, few approaches to embedding a DSML into a domain-specific modeling method (DSMM) that provides guidelines about how to use a given DSML and to evaluate related models. This chapter aims to make a contribution towards filling that gap by discussing, as an example and proof of concept, a domain-specific modeling method for the human cognitive modeling language HCM-L, a DSML for the domain of active and assisted living. As a modeling language without tool support has no chance to be used in practice, we are conducting that discussion on the basis

H.C. Mayr (✉) · F. Al Machot · J. Michael · G. Morak · S. Ranasinghe
V. Shekhovtsov · C. Steinberger
Application Engineering Research Group, Alpen-Adria-University Klagenfurt,
Klagenfurt, Austria
e-mail: heinrich.mayr@aau.at

F. Al Machot
e-mail: fadi.almachot@aau.at

J. Michael
e-mail: judith.michael@aau.at

G. Morak
e-mail: gert.morak@aau.at

S. Ranasinghe
e-mail: suneth.ranasinghe@aau.at

V. Shekhovtsov
e-mail: volodymyr.shekhovtsov@aau.at

C. Steinberger
e-mail: claudia.steinberger@aau.at

D. Karagiannis et al. (eds.), *Domain-Specific Conceptual Modeling*,
DOI 10.1007/978-3-319-39417-6_24

of HCM-L modeler, a tool that was implemented using the metamodeling platform ADO*xx* and can be accessed via OMiLAB, the Open Models Laboratory for modeling method engineering. HCM-L modeler is component of an ambient assistance system for supporting elder persons in mastering their daily life activities.

Keywords DSML design · Active and assisted living (AAL) · Modeling tool · Activity recognition · End user interaction

1 Introduction

Modeling and modeling methods are crucial for information systems sngineering. In practice, however, they are rarely embedded into all phases of development and operation: Practitioners challenge the benefits of modeling and complain about the confusing variety of concepts with overlapping semantics, symbols and syntactic rules of today's standardized, "universal" modeling languages.

Certainly, generic languages are meritorious due to their versatility in arbitrary domains as well as a broad body of experience and knowledge that has emerged from intensive use and research. Nevertheless, such languages tend to follow the "law of logistic growth" [1], being continuously extended up to a complexity and lack of concept orthogonality that affects their transparency and makes them hardly manageable for practical use. Think for example of UML which grew from initially five "diagrams" up to 17 (standard) and 8 additional diagrams in the version 2.0 [2]. Such complexity may lead to misunderstandings and scepticism.

As an alternative, domain-specific modeling languages (DSMLs) are gaining an increasing popularity: they are lean and convenient, support the productivity of modeling, and help to increase model quality and comprehensibility. Moreover, they come with lexical/graphical notations that are familiar and/or easy to understand by the users in that domain.

To increase the use of such DSML in practice, however, it has to be embedded into a domain-specific modeling method (DSMM), which features the procedure of how to apply the language, i.e., a "modeling procedure model" as well as appropriate mechanisms and tools to be used in such a procedure.

Few approaches have been reported so far regarding such efforts. This chapter, therefore, describes our results and experiences when developing a DSML for the domain of ambient and assisted living [3] within the framework of the project HBMS (Human Behavior Monitoring and Support).[1]

The aim of this project is to develop an AAL System, which

1. monitors an individual while carrying out daily life activities,
2. detects abstracts, aggregates and integrates the observed behavior into an individual human cognitive model (HCM), and

[1]This work was funded to a large extent by the Klaus Tschira Stiftung gGmbH, Heidelberg.

3. assists the individual in cases of need via a multimodal interface by retrieving knowledge from the human cognitive model from a case base and from a domain ontology using reasoning algorithms.

Thus, HBMS will facilitate it for elderly people with memory weaknesses to live longer autonomously in their familiar environment.

The chapter is organized as follows: Sect. 2 shows how DSMLs fit into the (meta) model hierarchy and describes current work on DSML creation processes. In Sect. 3, we outline the HBMS approach and present an overall architecture of the system. Section 4 is dedicated to the "Human Cognitive Modeling Language" HCM-L, which was developed within the HBMS framework. In Sect. 5, we discuss the modeling tool "HCM-L Modeler" which was developed using the metamodeling platform ADO*xx*. In Sect. 6, finally, we put the things together and describe the main HBMS functions. The paper concludes with a short outlook on future research.

2 Domain-Specific Modeling

A domain[2]-specific modeling language (DSML) is designed for exclusive use in a certain domain, and for specific purposes. When introducing a new modeling language, however, one should ascertain whether it is really needed or at least justified with respect to the intended application domain. As natural languages evolve over time following social, economic or environmental changes, modeling languages do so, too. They have to meet given challenges as efficiently and adequately as possible. Standardized languages have benefits due to their universal applicability and their wide range of concepts. However, exactly this wide range can be a drawback for the efficient and effective use of such a modeling language, in particular if non-experts—e.g., doctors, engineers or even the end-users themselves—should be able to understand and validate models intuitively. Consequently, in such cases a lean DSML, which comes with only few but appropriate concepts, may be justified.

Using the 4-level model hierarchy (see, e.g., [5–8]) as a basis, a DSML is an extension of M3 and a metamodel for M1 as shown in Fig. 1. That means that the DSML is defined on level M2 using a metamodeling language provided on M3. On level M1, the DSML is used to create concrete models that are instantiated on level M0.

Much work has been published on evaluating modeling languages [9, 10] on how to use a DSML [11, 12], but only few on the process of DSML/DSMM design.

Within the context of developing a DSML for enterprise modeling, [10] suggests a sequence of "macro process steps", namely clarification of scope and purpose, analysis of generic requirements, analysis of specific requirements, language

[2]A more detailed previous version of this section has been published in [4].

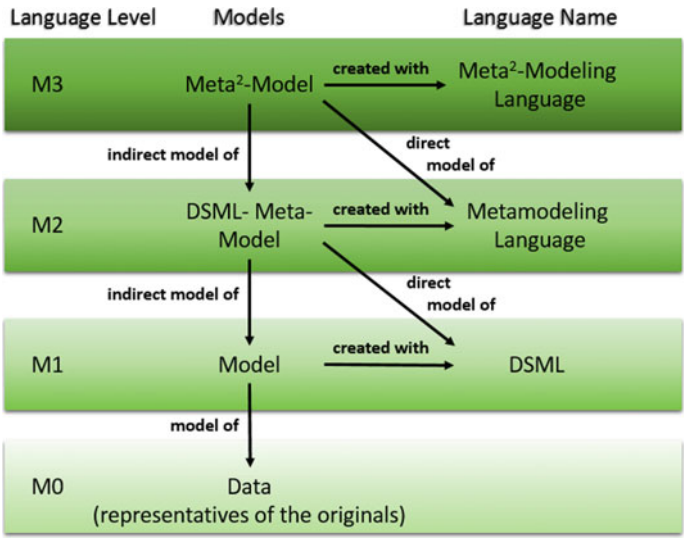

Fig. 1 Modeling hierarchy for a DSML

specification, design of graphical notation, development of modeling tool, evaluation and refinement. For each of these steps, several "micro process steps" are defined. In [11], modeling methods are defined as consisting of, (1) a modeling technique and (2) mechanisms and algorithms which work on the models (level M1). The modeling technique is divided into a modeling language, in our case a DSML, and a modeling procedure, which defines the application of the language. We propose an approach, which is based on this work, but is to some extent more generic than [10], and more reflecting domain-specific aspects than [11].

In particular, we propose to divide the DSML creation process in five main phases (see Fig. 2): *preparation*, *modeling language*, *modeling process, modeling tool* and *evaluation*. Each phase consists of several steps which are inspired by the work of [10] and [11]. These phases are only sketched here; for more details, see [4].

Preparation The preparation phase is to make sure that all relevant facts of the domain in question are known and well defined. We divide this phase in the steps *clarification of* scope and purpose, *requirements analysis* and *context analysis*.

Clarification: The *scope* drives the definition of the modeling concepts to be provided as part of the metamodel, which again determines the models that can be created on level M1. The *purpose* mainly relates to the profile of future user groups of the intended DSML: users of the modeling tool, users who have to understand the models on M1, e.g., doctors in the case of AAL.

Requirements Analysis: The main task here is to reveal the aspects to be modeled. This can be achieved by creating usage scenarios and exemplary diagrams. Another source of knowledge is domain-specific standards and relevant literature,

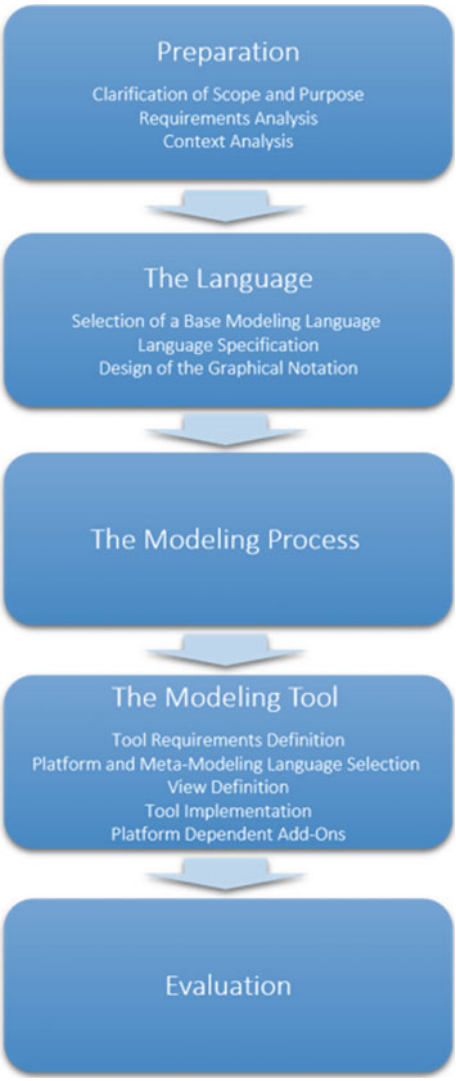

Fig. 2 The DSMM-creation process

and even more important: stakeholder involvement. Clearly, requirements analysis has to be done iteratively, until a stable specification has been reached.

Context Analysis: Although context analysis elicits requirements too, treating it as a step on its own may lead to a deeper understanding of the given domain. As an example, [13] introduces a context model focusing on a person's surroundings, such as things, services and information accessed by the person, mental and physical information about the person, social aspects like friends or relatives, context about what a person is doing and spatial–temporal information. A comprehensive overview of current context modeling approaches is presented in [14].

Language Creation The Language Creation phase concentrates on the language definition on level M2. First it has to be made clear, based on the results of the preparation phase, which modeling dimensions—structure (statics), function (operations) or behavior (dynamics)—have to be covered by modeling concepts. Also, to simplify the task, it is advisable to evaluate existing universal and/or specific modeling languages with similar scope and purpose, in order to select one as a basis from which the intended DSML can be derived. The language definition then is done by developing a metamodel and an appropriate graphical notation.

In general, a modeling language is defined by specifying its syntax, semantics and notation. For an overview of current approaches to syntax and semantics definition see, e.g., [11]. For specifying the graphical notation, the nine "*principles for designing cognitively effective visual notations*" as presented in [15] represent a good guide. However, also the idea presented in [16] is worth considering, namely to engage novices in designing symbols that are comprehensible to novices, as this could outperform experts' results. As some language constructs might be too complex for graphical representation (e.g., logical conditions), other appropriate forms of representation have to be defined. Again an iterative approach is necessary, informed by experiments involving the relevant stakeholders.

Modeling Process The modeling process definition should provide a stepwise procedure of how a particular model may be systematically built using the given DSML: which aspects should be modeled first, which view a modeler should start with (if there is more than one). The procedure should cover all modeling elements to provide a comprehensive insight for the modeler, and possibly also provide a "style guide" for which pattern to be used in which situation [17]. In Becker [18], some (general) useful rules are provided for creating business information models: to model only relevant parts, to leave irrelevant parts of the UoD, or to take care about naming conventions.

Modeling Tool For a newly created DSML there inherently is no ready-to-use modeling tool. Consequently, such a tool has to be created from scratch or by adapting an existing one. Again, several steps are to be performed in order to end up with an appropriate solution: (1) *tool requirements definition*, (2) *selection of a platform/framework and a metamodeling language*, (3) *view definition*, (4) *tool implementation* and (5) *platform-dependent add-ons*.

A framework for step 1 can be found in [17]. For step 2, we propose an approach opposite to that of [10]: instead of starting with selecting a metamodeling language, for economic reasons we would first select an appropriate framework/platform for tool generation. The decision then includes the choice of the metamodeling language. There are several such frameworks or platforms [11, 19].

As humans have perceptual and cognitive limits, it is important to provide different views on the content (step 3); these should help to reduce the complexity, e.g., by dividing a model into parts or by appropriate abstraction mechanisms focusing on particular aspects; also overall cognitive maps may help the reader to assemble information into a coherent mental representation of the models (see [15]).

The metamodel of the DSML should be formulated using the framework's metamodeling language. Based hereon, the tool implementation (step 4) is generative to the extent supported by the chosen platform.

Step 5 consists in exploiting—along the requirements—platform-specific features, e.g., interfaces or coupling possibilities to external software, components for model checking, simulation, analysis, transformation or generation of documentation.

Evaluation The evaluation has to be carried out in co-operation with the stakeholders against the goals and requirements, which have been determined in the preparation phase. Also, quality aspects on levels M1 and M2 have to be evaluated (for model quality categorization see, e.g., [20, 21]). A promising approach for completeness checks on level 2 is a pattern-based analysis like the one presented in [22] for business process modeling.

3 HBMS System: Architecture and User Roles

With advancing age, there is a tendency towards having an impaired memory and thus forgetting how to overcome the challenges of the daily life: "episodic knowledge" gets lost in parts.

Current forecasts of the global population ageing make clear that cognitive impairments are becoming a major problem in our societies. Thus, in 2050, more than 2 billion people will be over 60 [23], nearly 80 % of them living in the world's poorer countries [24]. This global challenge forces researchers around the world to focus on active and assisted living (AAL) (formerly: ambient assisted living [2, 20]), a subdomain of ambient assistance. AAL aims at developing methods, tools and software systems that enable elderly people, unobtrusively, to stay autonomous in their homes. AAL was pushed substantially in 2004, when it became a strategic support action (SSA) in the 6th Framework Program of the European Union, and since then, large budgets are assigned to the research into AAL and healthy ageing.

The aim of HBMS (human behavior monitoring and support) has been to maintain the personal autonomy of an elderly person as long as possible by supporting her/his individual mental processes. For that purpose, HBMS relies on conceptual behavior modeling, and on reasoning algorithms for deriving optimal support from an integrated model of abilities and episodic knowledge that an individual had (or has, but has temporarily forgotten). Broadly speaking, the HBMS process consists of observing the behavior of a target person (*learning mode*) using available activity recognition infrastructure and systems, and to transform and preserve the observations in a knowledge base, the *human cognitive model (HCM).*

In *support mode*, the HCM, a case base of concrete observations and a potentially existing domain ontology are exploited to assist the target person, when needed, in taking appropriate actions for reaching her/his current goal (Fig. 3).

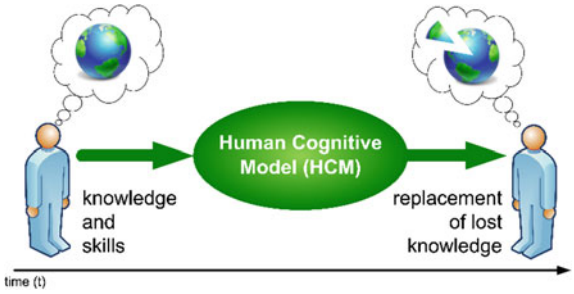

Fig. 3 Human cognitive model, kernel of the HBMS concept

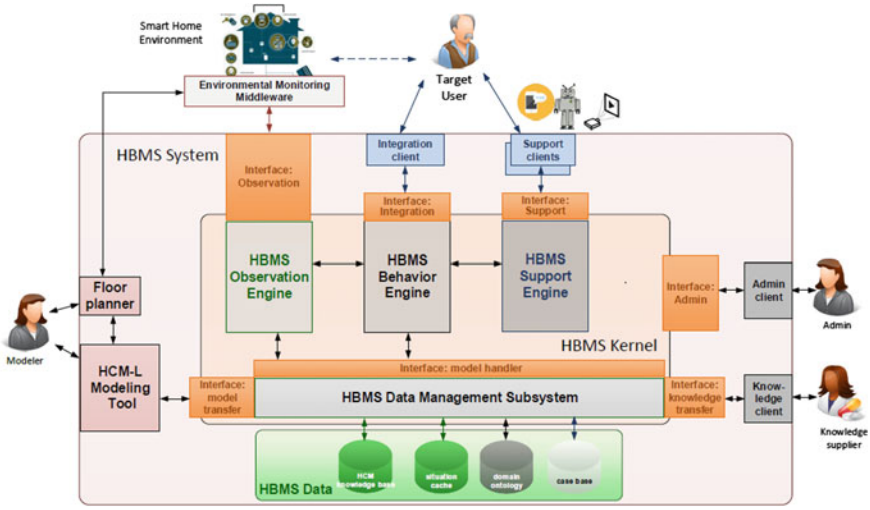

Fig. 4 Simplified HBMS-system architecture

Figure 4 shows a simplified architecture of the HBMS system. Activity monitoring and context acquisition is done outside the HBMS boundaries. We are working at a universal observation interface, which will allow to connect any available activity recognition or environment monitoring system to the HBMS system by automatically transforming their outputs into instances of HCM-L, the modeling language developed within the framework of the HBMS project. In the current stage, Nimbits [http://www.nimbits.com], an open source middleware tool for the Internet of Things, is used for this purpose.

There are four roles a HBMS-system user can take by applying the appropriate HBMS-clients:

Target User The target user resides in a smart home environment. The environmental monitoring middleware monitors his/her behavior using sensors or other activity recognition methodologies. Via the observation interface, the *HBMS-*

observation engine communicates to this environmental monitoring middleware. It listens to the observation data arriving from this component, analyzes and processes this data, and transfers recognized behavior situations to the *HBMS behavior engine*; the HBMS behavior engine handles the behavior situations arriving from the observation engine in context of the current HCM knowledge base. In learning-mode, the behavior engine collects obtained behavior situations to form a behavior sequence and integrates this sequence into the existing HCM. In cases of integration-doubts, the target user (or a deputy) is involved into the behavior integration process via an integration client.

In support mode, the behavior engine retrieves appropriate knowledge from HCM using reasoning mechanisms and transfers suitable knowledge chunks to *HBMS support engine*. The HBMS support engine is responsible for the context-sensitive multimodal assistance of the target user by using an appropriate support client.

Modeler Using the HCM-L modelling tool, the modeler is able to describe the behavior of the target user manually using HCM-L as graphical modeling language without using the learning mode capabilities of the HBMS-system described above. The HCM-L modeling tool is also the means to visualize, analyze, verify and validate HCM knowledge which has been learned while monitoring the target user. As an example, a doctor could also act in this role for diagnosing possible impairments of the target user. The floor planner tool facilitates the modeler to easily describe the environmental context of the target user in the form of graphical floorplans.

Administrator Via the admin client, the current state of the system can be monitored and visualized. Also, all administrative functions like user management or support client configuration can be dealt with using this client.

Knowledge Supplier The knowledge client allows to obtain knowledge from external sources and to integrate this knowledge into HBMS data (e.g., domain knowledge, external behavior knowledge).

In brief, the HBMS system includes the following data sources:

- *HCM*—the complete cognitive model of the target user; it serves as a source for generating situation cache elements,
- *HBMS Situation Cache*—the operational knowledge base in the system: it contains currently relevant and referenced HCM fragments, and state data collected from observations,
- *HBMS Domain Ontology*—an ontology representing domain-specific knowledge, which is referenced from both, HCM and HBMS Situation Cache;
- *HBMS Case Base*—a database of all observed action sequences; it is exploited, e.g., by reasoning algorithms of the support engine for weighting alternatives when determining the most likely next step to be advised.

4 HCM-L, a Modeling Language for the AAL Domain

The Human cognitive modeling language (HCM-L) is a lean modeling language which serves to represent and reproduce episodic knowledge of a certain person without loss. The scope is limited to the episodic knowledge of a person (autobiographical events and contextual information) and is further restricted to activities which were planned to be supported in the HBMS system, like activities of daily live, usage of electronic devices or software and others [25].

The terms *'cognitive modeling'* or *'models of cognition'* originate from cognition psychology and are important concepts for cognition science as well as for Artificial Intelligence (AI) [26]. In psychology and AI, theories of human problem solving are tested using a computer program, which tries to have the same control processes during problem solving as humans are supposed to have (see, e.g., the General Problem Solver in [27]). The main goal of cognitive modeling is to find out the basic principles of human intelligence. Usually, this is done by checking if a given AI-model is able to find solutions for a problem similar to supposed human problem-solving processes; or, in the opposite direction, psychological findings about the human memory are validated by simulation using AI-techniques [28].

Methodical considerations encouraged us to design and apply the method development approach as presented in Sect. 2. Concerning the *preparation phase*, the *clarification of scope and purpose* stemmed from the initial project idea which was motivated by personal experiences of one of the authors within his familiar environment. The *requirements elicitation and analysis* step started with a comprehensive literature analysis about the state of the art in AAL systems, which was followed by elicitation workshops with interested person of different age groups, as well as an analysis of common (generic) modeling languages. The latter revealed that these languages did only partly fulfil the elicited requirements. For the *context analysis* we carried a deep analysis of current AAL projects, research concerning smart homes, pervasive and ubiquitous systems, as well as activity and behavior recognition. As general standards for the domain are under development but rarely used in projects, e.g., universAAL [29], we could not use them as such.

As it was quite clear from requirements analysis that we would need a modeling language integrating dynamic (behavior) and structural (context) aspects, we decided to use previous experiences in conceptual modeling gained with KCPM, the Klagenfurt Conceptual Predesign model [30, 31], a lean, user-centred language for software requirements modeling, and use this as the basis for HCM-L. Consequently, the KCPM concepts were evolved and adapted to cognitive modeling. Thus, we could benefit from adopting the thing concept (a more intuitive way of abstracting from classes and attributes), as well as from the KCPM approach to relate resources and actions.

The HCM-L metamodel was created in several iterative steps using a UML class-diagram-like representation. As our goal was to create an intuitively understandable language, some of the (needed) complexity was displaced to a textual sub-language. For the same reason, it was decided to provide as few graphical

elements as possible. Some sample diagrams were created with this first draft set of modeling elements. The elements were revised in several iterations based on feedback of end users, colleagues and because of findings from [15], e.g., icons were added, the color and thickness of different connections was changed for a higher visual distance.

With HCM-L, the resulting "Human Cognitive Modeling Language", a new variant of the concept of cognitive modeling has been added, which is closer to the meaning of modeling in Informatics: The cognitive model is seen as an abstract extract of the episodic knowledge of a person; as such, it can be used as a knowledge base for services such as support activities, diagnosis, time series analyses and others.

In the realm of HBMS, models are encapsulated units of a certain person's behavior and of the respective relevant context. As such, HCM-L models form a knowledge base for reasoning services to optimally support a person: they are the core of the HBMS-System, and the central source of knowledge for other system components.

A preceding analysis of common (generic) modeling languages revealed that these only partly fulfilled our requirements [32]. As a consequence, we decided to create a DSML fulfilling the key requirements:

1. to provide models which can be used as a knowledge base in the support system,
2. to focus on human behavior and its context, and
3. to be understood intuitively by the relevant stakeholders in the AAL domain, e.g., people in general and their relatives, caregivers or doctors.

Consistent with the aim of the HBMS project, our DSML development was driven by the focus on conceptualizing human activities of daily life in the private home of one person, e.g., using electronical devices, dressing, cooking, writing an email and so on [4].

Following [11], the HCM-L *syntax* is described by a metamodel (see Fig. 5), the *semantics* by explanation in natural language, and the *notation* by a set of graphical elements (see Fig. 6). HCM-L is grounded in activity theory [33], which describes the nature of human activities in general.

The key modeling concept of HCM-L is *behavioral unit (BU)*; a BU is defined to encapsulate alternative sequences of individual actions a person may perform to reach a particular *goal*. For modeling actions, HCM-L features the concept *operation*; operations are connected by directed *flows*, which allow to represent action sequences and thus the direction of the behavioral process.

Operations may differ in detail (i.e., taking a coffee cup or a tea cup), in particular depending on the effects of a previous one. To describe such detailed functionality, HCM-L offers the concept *instruction*, as an attribute of operation. Basically, an instruction consists of simple additions, modifications and removals of concrete relationships between context elements (see below).

As there might be alternative sequences of actions for reaching the same goal that have one or more actions in common, there might be forks (i.e., several flows

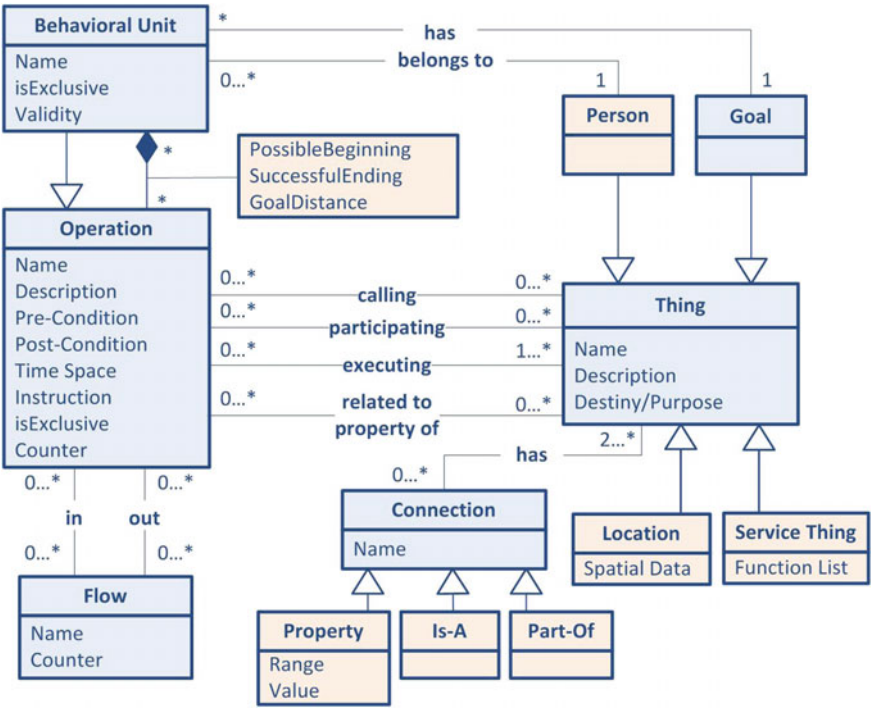

Fig. 5 HCM-L metamodel

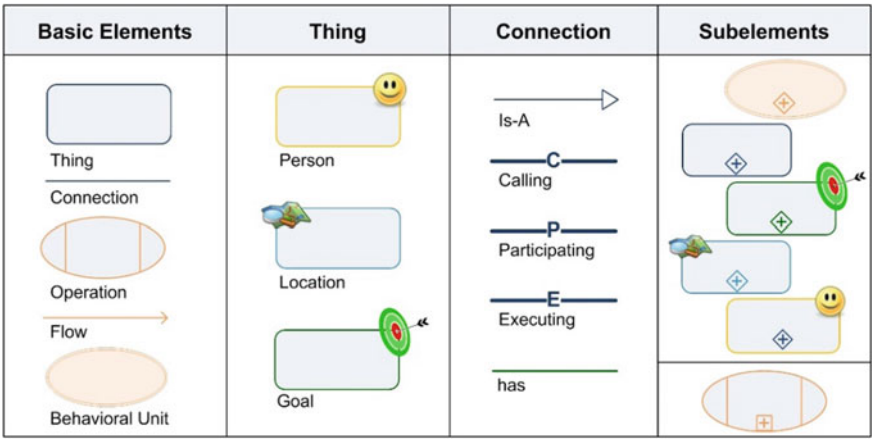

Fig. 6 Graphical notation of the HCM-L

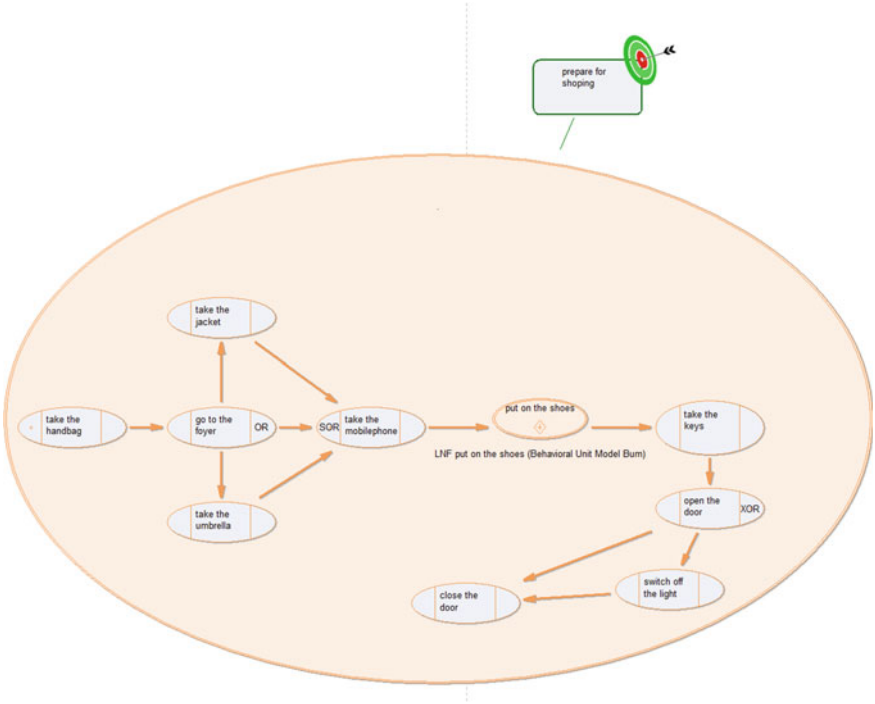

Fig. 7 Behavioral unit "prepare for shopping"

start from the same operation), and merges (i.e., several flows direct to the same operation), see Fig. 7. Consequently, modeling concepts are needed to describe the conditions that control the choice of the outgoing flow (and as such the correct sequencing) as well as the detailed instructions depending on the incoming flow. For this purpose, HCM-L features the concept of *Pre-* and *Post-Conditions*. Pre-Conditions define what conditions should be fulfilled before an operation is executed, and such may also influence the detailed instruction to be performed. Post-conditions define which operation should be executed as a next step, i.e., evaluate to a particular Flow. conditions may refer to values of attributes of things, related operations that have been successfully executed before, time constraints or combinations of all these. Pre- and post-conditions are formulated as logical expressions in infix-notation; except from their leading logical operator—AND, XOR, OR and SOR (synchronized or), see Fig. 7—they are not represented in the main graphical model but in notebooks related to each element and thus also to operations. This design decision reduced the semantic complexity of the graphical model. A formal language for defining more complex conditions and instructions was developed in the meanwhile and will be published separately.

To model the context in which daily human activities take place, HCM-L features concepts for all context dimensions, as defined by [13]:

- personal context like mental and physical restrictions of a person,
- environmental context like furniture or resources,
- social context like relatives or caregivers, and
- spatial–temporal context, such as location, time, frequency or duration of activities.

Operations are performed within such personal, environmental, social, and spatial–temporal context. Thus, each operation is connected to relevant concepts of the activities' context. For instance, an operation 'read the newspaper' would be connected to the newspaper itself, the person reading it, and a certain location where the newspaper and the reader are; it is performed every day at 7 am, which is preserved in an attribute of the operation.

One of the key demands on our DSML is the intuitive understandability of the models by our future users. Therefore, we had to design concepts and graphical representations, which are *intuitively understandable* for the *relevant user groups*: the supported persons, relatives or caregivers. We used research results about designing cognitively effective visual notations [15] to design a graphical notation that fits these needs, see Fig. 6. As shown in Fig. 6, each element corresponds to an element in the metamodel. Service-Things are currently not realized in the modeling tool. The validity of our assumption that these representations are intuitively understandable was proved by several experiments [34].

In accordance with activity theory, HCM-L observes the nature of human activities on different levels: (1) a BU as an overall process (represented by the BU symbol without content), (2) a BU with its operations and flows (represented by the BU symbol with content), and (3) a BU as a subtask, which means that a BU may inherit the properties of an operation (see Fig. 5), and, at the lowest level, (4) operations that realize actions [35]. Thus, it is possible to model the hierarchical structure of activities, which is one of the five principles of activity theory.

The element Operation-Makro (bottom right in Fig. 6) was introduced to summarize a sequence of operations without branches and merges by one model element. This helps in reducing the number of elements in a model and shortens the diagram. Thus, it supports human complexity management. In contrast to the other elements, the operation-Makro has no semantic meaning and no equivalent construct in the metamodel.

A *semantic analysis* of HCM-L [32] based on workflow patterns showed that it is—for the domain of ambient assistance—sufficiently powerful and fits the need of modeling human daily activities of one person better than general-purpose languages like BPMN or UML.

5 HCM-L Modeler, an ADO*xx*-Based Modeling Tool

HCM-L Modeler is a modeling tool for HCM-L including syntax, semantics and consistency support. It was developed using the metamodeling platform ADO*xx*, which implements the upper three layers of the OMG meta object facility (MOF).

ADO*xx* is a platform, which helps to build a full supported, professional and personalized modeling tool within a domain-specific application environment. It provides the possibility to develop syntax, semantic and graphical notations of any modeling concept based on different built-in functionalities. Furthermore, it features a scripting language that can be used to control the modeling elements, reference modes, consistency check, querying and many other functionalities. Additionally, it offers the execution and the coupling of external support modules that might be implemented using different programming languages, e.g., Java, R or C++.

HCM-L Modeler allows users to apply the HCM-L modeling concepts fast and easily. It controls the modeling process such that only syntactically correct models can be created. The models are displayed in such a manner that links to all related contexts proposed in HCM-L may be shown and checked.

HCM-L modeler can be used for different applications in the field of cognitive modeling and human behavior understanding. Figure 8 shows an overview of the HCM-L modeler functionality and file structure when modeling the BU of Fig. 7.

Figure 9 gives an impression of the context modeling screen, in particular "Maria's" context, elements of which are related to the operations in the BU "prepare for shopping" as shown in Fig. 7. We are aware that, given the layout rules for this volume, details are hardly readable.

We now shortly outline the major functionalities supported by HCM-L modeler; for details, see [34] or visit the handbook in [36].

Visualization Support All elements and relation classes can be connected in a flexible way. The breaking lines of elements names can be resized autonomously. This flexibility is an advantage in cases where visibility is required to make the model better readable by humans. Furthermore, the modeling area can be zoomed in and zoomed out, and the model navigator helps to track the location of the element in the drawing area.

Model Stepper The stepper animates the succession of operations (of the active model) and allows a stepwise pass through a behavioral unit based on users' decisions. This functionality is supported to step even within referenced sub-models. Basically, this is achieved by highlighting the visited operation. By tracking the operation flows, the stepper supports model understanding and validation. Currently, the stepper steps autonomously, if the operation has one outgoing flow, otherwise, the user gets a dialogue window which lets him choose the next path.

Autonomous Referencing Essentially, the tool creates reference models autonomously and does not allow redundant reference models. This feature supports consistency by helping users to create models without contradictions. Additionally, the attributes of each model element can be specified easily after clicking twice, using the mouse to explore the particular notebook. Various types of

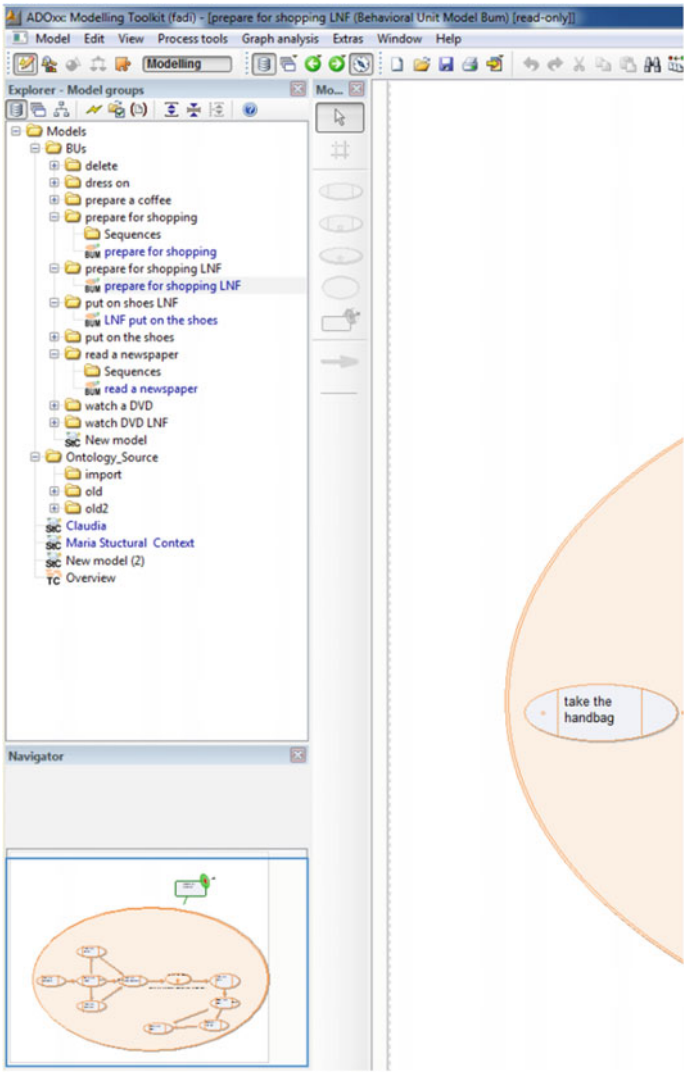

Fig. 8 HCM-L modeler screenshot with BU of Fig. 7

attributes may be added, e.g., enumeration lists, radio buttons, text fields, browse buttons. Autonomous referencing is supported also when adding a relation class, a reference model using the notebook.

Querying The AQL query language is a built-in querying language provided by the ADO*xx* platform. It allows querying models in a style similar to SQL. In ADO*xx*, queries can be pre-defined by the developer or may be formulated manually by a user. The current version of HCM-L modeler does support manual queries as shown in Fig. 10.

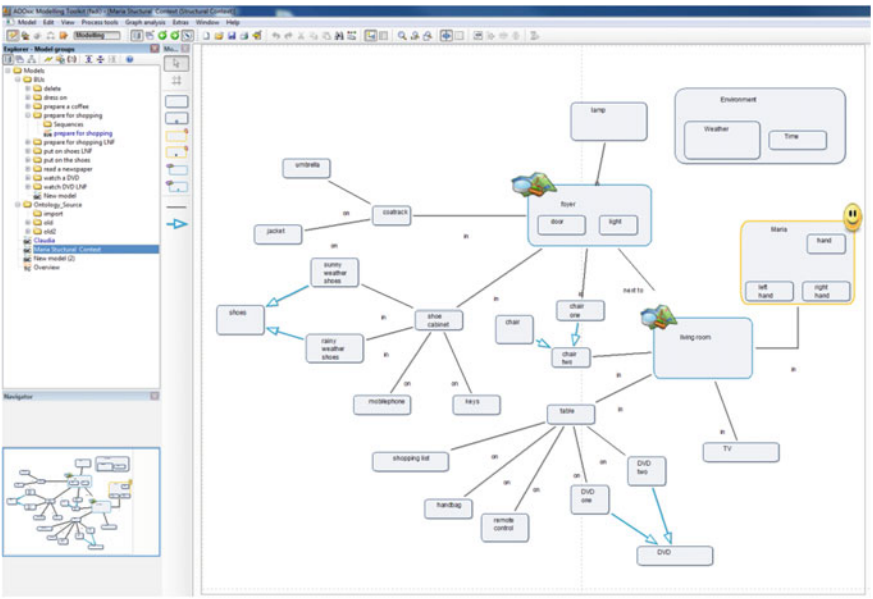

Fig. 9 Context modeling screen

Consistency check The HCM-L modeler is designed to be user-friendly for modelers in order to facilitate the development of correct models: no advanced modeling skills should be needed for creating HCM-L models on level M1. To extend the consistency check feature for particular needs, regular expressions can be added by an expert to control inputs into notebook fields.

Importing and exporting models ADO*xx* offers the possibility to import and export models in a generic XML format. This feature is adopted by HCM-L Modeler in order to allow transforming models to other formats like XML or OWL (implemented within the realm of the HBMS project), as used, for example, by inference or reasoning tools.

Reasoning Support Both model- and rule-based reasoning approaches for behavior modeling require the extraction of different features out of the given overall model. The HCM-L Modeler, among others, offers the possibility to calculate the frequency of specific activities based on the user history: every operation is supported by a percentage value. Furthermore, it delivers for every operation the smallest number of the remaining operations (i.e., operations to be executed) until reaching the current BU's goal, together with all possible paths leading to that goal and under consideration of all subunits.

Media Files HCM-L Modeler offers the possibility to upload media files (video, audio and images files in different formats) into the tool. This feature allows using such files for visualizing complex issues and situations in the support phase.

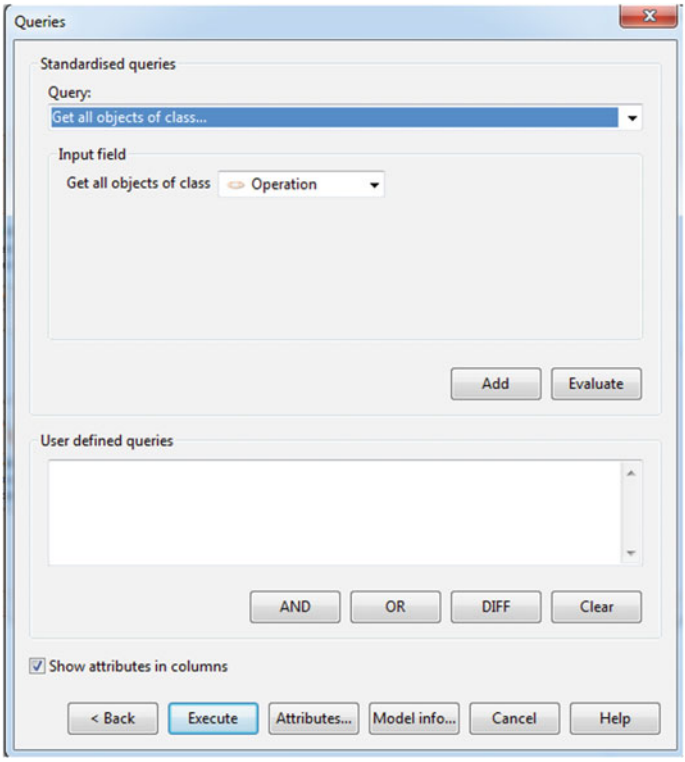

Fig. 10 Querying support in HCM-L modeler

To sum up, ADO*xx* proved to be an appropriate platform for implementing a HCM-L modeling tool. The HCM-L modeler is part of the HBMS system (see Fig. 4), and allows to feed manually created models into the knowledge base, as well as to visualize and analyze models that are automatically generated via activity recognition and the HBMS observation engine.

The next release of HCM-L modeler will provide an advanced rule support dialogue window, which will help the expert user to add rules correctly.

6 Putting Things Together: The HBMS Functionality

6.1 Coupling Activity Recognition Systems

Recent advances in computer science and engineering, electronics and sensor technologies have resulted in a significant progress in the field of designing and implementing activity recognition (AR) systems. Most of these systems are designed to fulfil specific goals, i.e., without the intention of integrating them into

more comprehensive ambient assisted living (AAL) environments. Consequently, coupling arbitrary AR systems to HBMS is a challenging task, especially from the point of view of meeting user expectations, and guaranteeing the fulfilment of HBMS functional and non-functional goals continuously during lifetime.

Coupling AR systems, however, would come with many potential advantages [37]:

- Multiple AR systems provide redundancy to guarantee uninterrupted supply of human behavior information, even in case of the failure of one or several AR systems.
- As a rule, a single AR system can only provide a specific subset of observation or context data ("opening a cupboard", "watching TV"); in this case, coupling such AR systems together will enhance the provided coverage of the environment (e.g., spatial and geometrical coverage) as the information hidden from one system can be completed by the others.
- Integrating AR systems would increase the accuracy and confidence of observation by augmenting the observation data coming from one AR with the information obtained by the others, and by validating such data by comparing it to the data coming from other connected AR systems.
- Obtaining the information from coupled AR systems reduces the ambiguity of interpretations, and hence decreases the level of uncertainty in the data. This, in turn, improves object detection.

In general, the AR process consists of several steps: from raw sensor data collection to the activity recognition (Fig. 11). Initially, raw sensor data is collected and passed through the data pre-processing phase to remove the noise and the redundant data from the raw sensor stream. Then, the pre-processed data passes through the segmentation phase to identify the most useful segments of the data. These segmented data are used in the feature extracting phase to extract the main features of the data. The feature extraction phase is followed by a dimensionality reduction process to increase the accuracy of the features and to reduce the computational effort needed for feature classification. Finally, the selected feature sets are used for feature classification and for the recognition process [38].

The traditional activity recognition phases allow for identifying simple human activities. Modern AAL systems, however, require the recognition of more complex activities in order to be able to provide better support. However, identifying complex human activities is challenging. The current approach to solving this problem is augmenting the behavior data with context data of different kind, i.e., environmental, spatial, and temporal data, and trying to predict more complex human behavior patterns based on this integrated data.

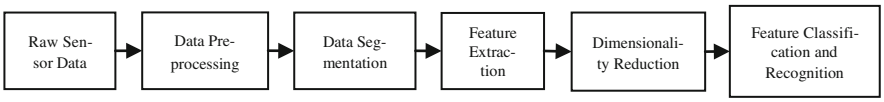

Fig. 11 Activity recognition phases

In order to retrieve more context data, AR systems need to process more sensor data coming from the environment. Most of these sensors provide different kinds of heterogeneous data. Consequently, introducing a common interface for importing and interpreting such data (AR system interface, see Fig. 4) is an essential task. Such interface should be capable of both, handling incoming heterogeneous data and transforming them to units that are processable by the observation engine. Therefore, within the HBMS framework, following the approach introduced in [39], a metamodel for a domain-specific graphical modeling language (AREM-L—Activity Recognition Environment Modeling Language) is under development. The AREM-L generic model and a site-specific AR system configuration are separated by means of an AR system workspace concept, which incorporates both, the generic model and the necessary configuration information.

There are several approaches to implement such AR system interface. First, the interface could be placed before the data pre-processing step, e.g., as a layer which collects raw sensor data from various heterogeneous sensor devices and converts it into a generic homogeneous data format prior to sending for pre-processing.

Second, the interface could be placed at the end of the chain of activity recognition phases (Fig. 12), e.g., as a layer which collects all simple events and context data recognized by the AR systems, and performs a data fusion process to identify complex human behavior activities out of the collected data.

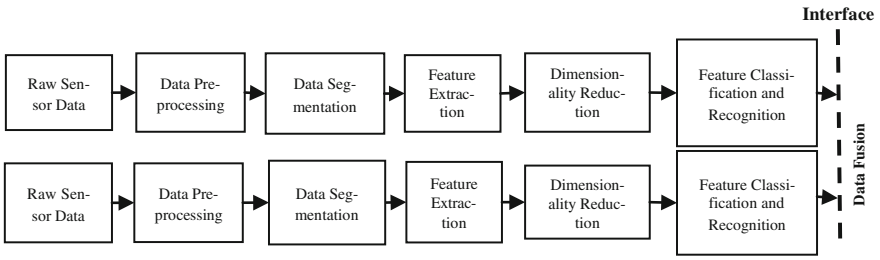

Fig. 12 AR system interface located after the (simple) activity recognition step

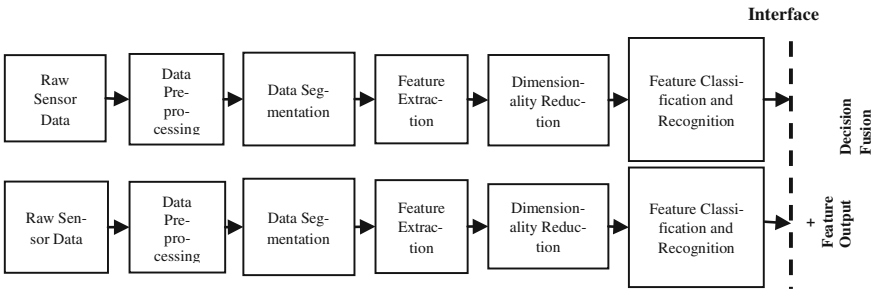

Fig. 13 Decision fusion interface

Third, the final output could be produced by combining several decision support methodologies involving experts (such as voting, fuzzy logic, and statistical methods) to enhance the prediction accuracy (see Fig. 13) [37].

In the HBMS system, we combine the second and the third approach: Coupled AR systems are supplying their pre-processed sensor data; these are transformed in compliance with our AR system integration interface such that both behavior and state context data form HCM-compliant data structures.

6.2 Model-Based Reasoning

Model-based reasoning refers to an inference approach which is applied to a domain-specific model. The major step in model based reasoning is to create first the model, and then, at run-time, run a "smart engine" which combines the knowledge that is provided by the model and the observed sensor data to derive conclusions such as a diagnosis or a prediction.

In the HBMS system, this knowledge and the imported activity recognition data are represented as instances of an HCM-L model. Two reasoning modules work with this data, *prediction module*, and the *rule-based reasoning module*. The *prediction module* is used to find the most likely next operation, when the user stops and does not perform any new operation, or when the currently "running" Behavioral Unit has to be determined. The *rule based reasoning module* is used to find the operation the observed person actually is performing, and to identify the BUs the operation is part of.

The Prediction Module In smart home environments, it is common that different activities share many similar or identical sensors. As an example, the prepare-a-meal and prepare-a-drink activities share simple events like enter the kitchen, open the cupboard and open the fridge. Thus, such situations form a kind of uncertainty which causes poor or bad decisions.

Therefore, the prediction module is needed to predict (1) what the current BU is and/or (2) to predict what the next possible operation is when the model contains more than one possible flow from the current operation.

The key point of our reasoning module is to specify a reasonable window size which is based on the optimal N sensors of the provided environment. Choosing optimal sensors is based on processing the training data using an SVM classifier [40]. In this approach, attributes are ranked by the square of the weight assigned by the SVM. Attribute selection for multiclass problems is processed by ranking attributes for each class separately using a one-vs-all technique and then "dealing" from the top of each pile to give a final ranking [40], i.e., in HBMS: the optimal number of sensors that best characterize this activity.

For (1), after having determined the optimal sensors, the sensor data are collected until the optimal sensors of one activity are activated. As soon as this

happens, the following three steps are applied: (a) the assignment of priority levels to sensors, (b) the adjustment of sensors belief and (c) the evidence combination of optimal sensors beliefs using an advanced version of Dempster–Shafer rule that delivers logical results even in the presence of high conflicting data [41]. The evidence combination is performed with respect to the ordered set of sensors priorities that are provided using the optimization capabilities of answer set programming.

For (2) the same mechanism is applied but using the reasoning attributes of the current operation. Thus, we have an optimization problem to be solved based on three priority measures: (1) the importance of performing an operation according to the user history; (2) the cost value of choosing an operation based on the similarity between the current user profile and other users; (3) the time when the operation should be performed [42].

The Rule-based reasoning module Another situation where reasoning is needed is to query the HCM-L model with respect to the logical expressions in the pre- and post-condition slots of each operation. As an example, think of the condition that a specific medicine has to be taken 30 min before breakfast.

SPARQL Inferencing Notation (SPIN) [43] has been selected for being used in HBMS for reasoning in such situations. SPIN supports inferencing over Ontology Web Language (OWL) texts, and as such offers a way to define and to apply constraint checking under closed world semantics and to automatically retrieve the appropriate answer from the sent queries with respect to those constraints. The advantages of SPIN are as follows: (a) it stores SPARQL queries with the model, (b) constraints can be natively executed, (c) it uses a simple form of backward chaining, and (d) it allows for computing sub-queries on demand.

As a result, the HBMS system, by combining optimization and rule-based methods, offers powerful reasoning even under uncertainty.

6.3 End-User Interaction

The main idea of the HBMS project is to support people in performing daily activities when cognitive functions, such as memory, are decreasing. Thus, after having recognized what a person is doing or not, and after having determined whether some advice is necessary, and if so, the best fitting advice has to be communicated to the person. Consequently, the challenge was to select devices (e.g., smart phone, tablet, audio and video devices, others) and to develop user-interfaces that older people can use intuitively and independently.

In a first prototype implementation, Microsoft ASP.NET MVC 3 framework and Microsoft Visual Studio 2013 as IDE was chosen for implementing an interface for tablets and smart phones. In addition to default libraries and packages of ASP.NET MVC 3, many other free and open sources JavaScript libraries such as jQuery

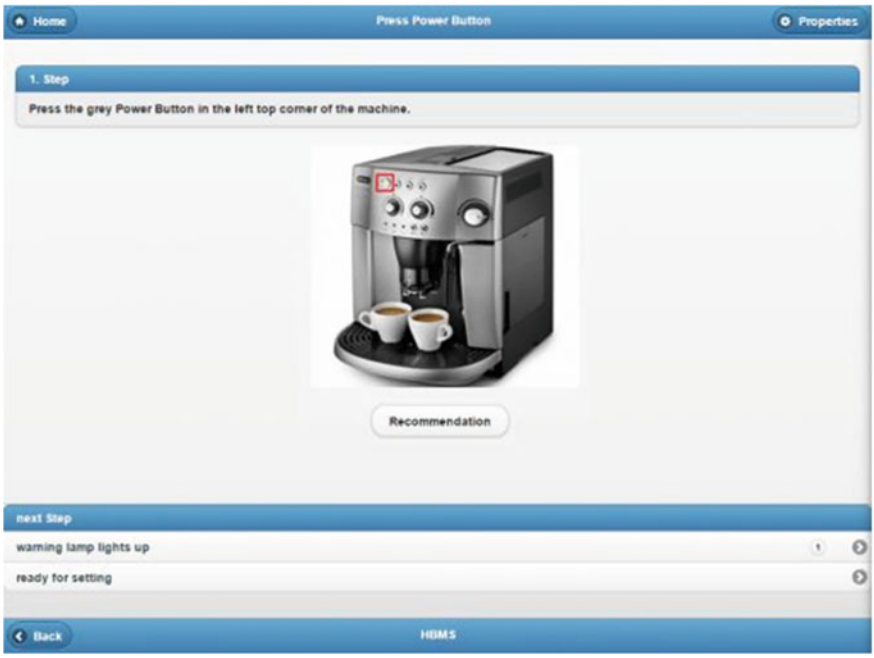

Fig. 14 HBMS interface prototype 1 (obviously, font sizes may be adjusted)

Mobile, jQuery mobile UI [44] and Ajax were used. The main reason for choosing ASP.NET MVC 3 framework was its MVC-based (model-view-controller) pattern and object-oriented aspect of C# programming language, which makes it reasonably easy to build standard and scalable Web applications based on the MVC paradigm. By employing jQuery Mobile components, it was possible to achieve an optimal adjustment to the *"Designing User Interfaces for Older Adults"* challenge, without impacting the functionality. In order to ensure independence from the system environment with regard to databases, the Open Source Persistence ORM Framework NHibernate was used, which is compatible with virtually all relational-database systems. This means that the HBMS-web application can easily be integrated into and used within any relational-system-environment.

Figure 14 shows a typical HBMS prototype support screen. The navigation buttons ("Home", "Back") are located on the top/bottom of the screen. In the middle, a short textual description of the current action with a picture or video is shown. Above the bottom, the possible next steps are listed. When pushing the recommendation button, short pieces of advice are presented in a frame at left-hand side.

In the current project phase, in addition to further development (e.g., improved interaction and multimodal-interface based on Web 2.0) of the existing prototype, we are going to support the output via a voice-command-system, using autonomous robots (e.g., Nao, Pepper [45]), that, in addition, can be used for sending sensor and camera data for activity recognition.

7 Outlook

HBMS is a running project. Prototypes have been released at different stages and presented to and explored by the interested public on various occasions in order to gain an impression of the viability and acceptance of the overall concept. Also, a laboratory was installed for performing experiments under laboratory conditions.

Within the current project phase, we are working, among others, on the following (partly already completed) tasks:

- Definition and implementation of the AREM-L (see Sect. 6.1) including the transformation of AR output streams to HCM-L Operation sequences including context references,
- Specification and implementation of the expression and instruction language as a part of HCM-L,
- Automatic integration of sequences into new or existing BUs; together with the previous task, this will accomplish an automatic HCM-L model generation from AR system outputs,
- Inferencing goal specifications from observed activities by exploiting domain ontologies,
- Integrating various device types for multimodal interaction (see Sect. 6.3),
- Performing first experiments in a real life environment.

We are aware of the fact that there is still a long way to go for having available a HBMS "mass product" which features all necessary means for activity recognition, is easy to roll-out and can be distributed at an affordable price. Nevertheless, we are optimistic and confident that we are on the right track.

Tool Download http://www.omilab.org/hcml.

References

1. Feller, W.: On the logistic law of growth and its empirical verifications in biology. Acta. Biotheor. **5**(2), 51–66 (1940)
2. UML 2.5 Diagrams Overview. http://www.uml-diagrams.org/uml-25-diagrams.html. Accessed 12 Mar 2016
3. ACTIVE AND ASSISTED LIVING PROGRAMME—ICT for ageing well. http://www.aal-europe.eu/. Accessed 12 Mar 2016
4. Michael, J., Mayr, H.C.: Creating a Domain Specific Modelling Method for Ambient Assistance. In: International Conference on Advances in ICT for Emerging Regions (ICTer2015). IEEE (2015)
5. International Organization for Standardization: Information Technology—Information Resource Dictionary System (IRDS) framework. ISO/IEC 10027:1990
6. Object Management Group: Meta Object Facility (MOF) Specification. www.omg.org/cgi-bin/doc/?formal/02-04-03.pdf. Accessed 12 Mar 2016
7. Atkinson, C., Kühne, T.: Model-driven development: a metamodelling foundation. IEEE Softw. **20**(5), 36–41 (2003)

8. Bézivin, J.: On the unification power of models. Softw. Syst. Model., **4**(2), 171–188 (2005)
9. Frank, U.: Domain-specific modelling languages: requirements analysis and design guidelines. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) Domain Engineering, pp. 133–157. Springer (2013)
10. Frank, U.: Outline of a method for designing domain-specific modelling languages. ICB-Research Report 42. University Duisburg-Essen (2010)
11. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) E-Commerce and Web Technologies. LNCS, pp. 182–197. Springer (2002)
12. Tolvanen, J.-P., Kelly, S.: Defining domain-specific modelling languages to automate product derivation: collected experiences. In: Hutchison, D., et al. (eds.) Software Product Lines, pp. 198–209. Springer (2005)
13. Kofod-Petersen, A., Mikalsen, M.: Context: representation and reasoning. Spec. Issue Rev. d'Intell. Artif. "Appl. Context-Manag." (2005)
14. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
15. Moody, D.L.: The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. **35**(6), 756–779 (2009)
16. Caire, P., Genon, N., Heymans, P., Moody, D.L.: Visual notation design 2.0: towards user comprehensible requirements engineering notations. In: IEEE 21st International Requirements Engineering Conference (RE), pp. 115–124 (2013)
17. Kaschek, R., Mayr, H.C.: A characterization of OOA tools. In: IEEE International Symposium on Assessment of Software Tools, pp. 59–67 (1996)
18. Becker, J.: Die Grundsätze ordnungsmäßiger Modelierung und ihre Einbettung in ein Vorgehensmodel zur Erstellung betrieblicher Informationsmodele. Informationssystem-Architekturen, Rundbrief des Fachausschusses 5.2 der Gesellschaft für Informatik **5**(2), 56–62 (1998)
19. Steinberg, D.: EMF Eclipse Modelling Framework. The Eclipse Series. Addison-Wesley, Upper Saddle River, NJ (2009)
20. Krogstie, J., Lindland, O.I., Sindre, G.: Defining Quality Aspects for Conceptual Models. Chapman & Hall, London (1995)
21. Batini, C., Ceri, S., Navathe, S.B.: Conceptual Database Design. An Entity-Relationship Approach. The Benjamin/Cummings Series in Computer Science. Benjamin/Cummings, Redwood City, Calif (1991)
22. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distrib. Parallel Databases **14**(1), 5–51 (2003)
23. United Nations Population Division, World Population Prospects, the 2012 Revision, Indicator Population 60+, World, medium variant. http://esa.un.org/wpp/unpp/panel_indicators.htm. Accessed 12 Mar 2016
24. United Nations, Department of Economic and Social Affairs, Population Division. World Population Ageing 2013. ST/ESA/SER.A/348 (2013)
25. Michael, J.: Using Cognitive Models for Behavioural Assistance of Humans. In: it—Information Technology, de Gruyter. Accepted 2 Dec 2015
26. Cohen, P.R., Feigenbaum, E.A.: The Handbook of Artificial Intelligence. William Kaufmann, Inc. (1982)
27. Newell, A., Simon, H.A.: Human Problem Solving. Prentice-Hall, Englewood Cliffs, N. J. (1972)
28. Schmalhofer, F.: Constructive Knowledge Acquisition: A Computational Model and Experimental Evaluation, Lawrence Erlbaum Associates (2001)
29. Ferro, E, Girolami, M., Salvi, D., Mayer, C., Gorman, J., Grguric, A., Ram, R., Sadat, R., Giannoutakis, K.M., Stocklöw, C.: The UniversAAL Platform for AAL (Ambient Assisted Living). J. Intell. Syst. (2015)

30. Kop, C., Mayr, H.C.: Conceptual predesign bridging the gap between requirements and conceptual design. In: Proceedings of the 3rd International Conference on Requirements Engineering (ICRE'98), Colorado Springs, April 1998, pp 90–100 (1998)
31. Mayr, H.C., Kop, C.: A user centered approach to requirements modelling. In: Glinz, M., Müller-Luschnat, G. (Hrsg.): Modelierung 2002—Modelierung in der Praxis—Modelierung für die Praxis, pp 75–86 (2002)
32. Mayr, H.C., Michael, J.: Control pattern based analysis of HCM-L, a language for cognitive modelling. In: International Conference on Advances in ICT for Emerging Regions (ICTer2012), pp. 169–175. IEEE (2012)
33. Leont'ev, A.N.: Activity, Consciousness, and Personality. Prentice-Hall, Englewood Cliffs, NJ (1978)
34. Michael, J., Al Machot, F.; Mayr, H.C.: ADOxx based tool support for a behaviour centered modelling approach. In: Proceedings of 8th International Conference on Pervasive Technologies Related to Assistive Environments PETRA 2015. ACM Digital Library Proceedings (2015)
35. Michael, J., Mayr, H.C.: Conceptual modelling for ambient assistance. In: Proceedings of 32nd International Conference on Conceptual Modelling—ER 2013. LNCS, vol. 8217, pp. 403–413. Springer, Berlin/Heidelberg (2013)
36. OMiLAB: HCM-L download. http://www.omilab.org/web/hcm-l/download
37. Raol, J.R.: Multi-Sensor Data Fusion with MATLAB®. CRC Press (2009)
38. Krishnan, N., et al.: Recognition of hand movements using wearable accelerometers. JAISE **1** (2), 143–155 (2009)
39. Shekhovtsov, V.A., Mayr, H.C., Kop, C.: Facilitating effective stakeholder communication in software development processes. In: Nurcan, S., Pimenidis, E. (eds.) Information Systems Engineering in Complex Environments. LNBIP, vol. 204, pp 116–132. Springer (2015)
40. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1-3), 389–422 (2002)
41. Ali, T., Dutta, P., Boruah, H.: A new combination rule for conflict problem of Dempster-Shafer evidence theory. Int. J.Energ. Inf. Commun. **3**(1), 35–40 (2010)
42. Al Machot, F., Mayr, H.C., Michael, J.: Behaviour modelling and reasoning for ambient support: HCM-L modeller. In: Modern Advances in Applied Intelligence, pp. 388–397. Springer International Publishing (2014)
43. Fürber, C., Hepp, M.: Using sparql and spin for data quality management on the semantic web. In: Business Information Systems. Springer Berlin Heidelberg (2010)
44. jQuery Mobile UI: https://jquerymobile.com/, accessed December 25, 2015
45. https://www.aldebaran.com/en. Accessed 25 Dec 2015