Control Pattern Based Analysis of HCM-L, a Language for Cognitive Modeling

Heinrich C. Mayr, Judith Michael

Department for Applied Informatics/ Application Engineering
Alpen-Adria-Universität Klagenfurt, Austria
{heinrich.mayr, judith.michael}@aau.at

***Abstract* – HCM-L, a conceptual language for modeling human behavior within the context of ambient assistance is introduced. By exhibiting the results of a pattern-based analysis it is shown that HCM-L features all concepts to express control flows as discussed in [1] and as is relevant for human behavioral modeling. The work is part of the Project HBMS (Human Behavior monitoring and support) which aims at supporting cognitive performance of individuals.**

***Keywords* - Innovative Applications, Research Challenges, Behavior Modeling, Pattern, Language Analysis, Ambient Assistance**

I. MOTIVATION

The evolvement of UML as a global standard for conceptual modeling did not reduce the continuous emergence of new conceptual modeling languages. These languages are mostly dedicated to a specific application domain or focus on particular requirements like user centeredness, minimalism, completely defined formal semantics and other aspects.

Clearly, when defining a new modeling language for a given purpose it should be analyzed and proven against both: (1) does the language feature the desired properties and (2) is it complete in the sense that it covers all aspects to be modeled within the targeted application domain.

Rizzi argues in [2] that patterns are of increasing interest in information systems research and are "*considered to be an effective answer to the advanced analysis requirements emerging in complex and scientific applications*".

A comprehensive taxonomy of workflow patterns for the analysis of workflow or business process modeling languages can be found in [3]. It is the result of an on-going joint initiative of Eindhoven University of Technology (W.M.P. van de Aalst) and Queensland University of Technology (A.H.M. ter Hofstede). Wohed et al. [4] show by the example of analyzing UML Activity Diagrams that these patterns provide a sufficient level of granularity for a deep analysis of process modeling languages and for evaluating their expressive power.

This paper discusses the analysis of HCM-L, a conceptual language for human cognitive behavior modeling based on the workflow pattern framework [3]. It shows that HCM-L features modeling concepts covering all those behavioral aspects that are relevant for cognitive support as is intended by the HBMS[1] (Human Behavior Monitoring and Support, [5]) research project. HBMS aims at supporting people by their own possibly lost or forgotten knowledge. For that purpose a Human Cognitive Model (HCM) is established by observation, modeling and integration processes. When needed, tailor-made support information is derived from HCM by reasoning processes.

HCM-L is based on concepts of the user-centered Klagenfurt Conceptual Predesign Model (KCPM [6], [7]) which were adapted to better reflect the structure [8] and contexts [9] of human behavior.

The paper is organized as follows: Section II introduces the human cognitive modeling language HCM-L and defines it semantics by a meta-schema. Section III outlines the basic notions of pattern analysis theory. In Section IV we present a pattern based analysis of HCM-L with a focus on control flow patterns. Section V briefly summarizes the work presented and outlines some research questions to be attacked in near future.

II. HUMAN COGNITIVE MODELING LANGUAGE

People need support in their everyday life when cognitive functions like memory are failing or decreasing. This might happen when we are stressed or unpracticed in doing specific things or when we are getting older and our brain is suffering from neurobiological changes. HBMS intends to support people by retrieving forgotten knowledge about activities ("*behavioral units*") of their everyday life. For "learning" that knowledge it is a good opportunity to build a cognitive model of a person's behavior while she or he is of sound mind and memory.

Sequences of personal actions may vary over the time: they may depend on special contexts or result from individual decisions for behavioral changes, omitting particular actions or rearranging the sequence of actions. A useful cognitive modeling language should make it possible to derive all correct variants (instances) of a behavioral unit from a generalized model of that unit. This mechanism is similar to that of Case Based Reasoning, where case prototypes are built as abstraction of similar cases [10].

HCM-L is designed for modeling human behavioral units. Consequently it offers concepts for the three modeling dimensions [11]: Structure (Statics), Function (Operations) and Behavior (Dynamics). This comprehensive approach was chosen in opposite to workflow modeling languages like e.g. BPMN [12] in order to provide a uniform and self-contained language and to profit from the user centered language concepts of KCPM. This is especially important in view of the intended users so that particular attention has to be spent to the intuitive understandability of the modeling language: this will allow the users to validate the models of their behavior more easily if they want to do so.

We now proceed to present the key HCM-L concepts. A quick overview is given by the meta-schema in Fig. 1.

A. Behavioral Unit, Operation and Condition

The *Behavioral Unit* is an aggregate of actions (*Operation*) that are interrelated via pre- and post-conditions. It is there for modeling everyday life activities that a person performs to achieve a certain goal. An example is the brewing of coffee in a coffee machine. The behavioral unit 'Coffee brewing' includes a set of actions, which follow each other step-by-step: take a cup, place the cup, choose the water and coffee volume

[1] Funded by the Klaus Tschira Stiftung, Heidelberg

978-1-4673-5530-8/12/$31.00 ©2012 IEEE

and press the start button. The decision on granularity, i.e. what to model as a behavioral unit and what as an operation, depends on the needs and views of the concrete situation. An indication for the decision can be gained from taxonomies as discussed in [13] (*Basic Activities of Daily Living, ADL)* or [14] (*Instrumental Activities of Daily Living, IADL)*. The latter focus on situations when a person uses technical devices or electronic processes, like make a phone call, shopping, meal preparation, housekeeping, laundry, medication and eBanking.

The semantic relatedness between behavioral units and operations is covered by the generalization between them: Using that concept, a behavioral unit can be used as an operation of a more comprehensive activity (behavioral unit): To 'book a flight' might be an operation in the behavioral unit 'plan a conference journey'. To plan this journey may also be a part of the behavioral unit 'participation at ICTer 2012'; other operations are to write and submit a paper, get informed about the acceptance, rework the paper and submit it again, register and pay the registration fee before you plan this journey and several next steps like to go on the journey and the conference attendance.

The execution of an operation may depend on one or more *Pre-Conditions* to be fulfilled; for example, we need to receive the acceptance notification for a submitted paper before forwarding the travel application to the administration. *Post-Conditions* describe the consequences of an executed operation.

Two operations within a behavioral unit are interrelated, if they have matching conditions, e.g. post (op_1) = pre (op_2). This concept corresponds to that of Petri nets and thus allows for modeling causal dependencies and independencies (concurrency) of operations.

There might be simple and complex logical dependencies between conditions; e.g., you might attend a conference if your paper is accepted, if you give a tutorial, or if you simply want to follow the talks of others. Such logical dependencies are modeled in the *Pre-Condition Expression* or *Post-Condition* Expression of the involved Operation. Clearly, the condition itself also may be defined as a logical expression. For the purpose of this paper, we do not go into details of the formal language for these expressions but defer this to a subsequent paper.

B. *Thing and Connection*

As pointed out before, HCM-L is comprehensive in the sense that it provides concepts for the fundamental modeling dimensions. Regarding the static dimension we adopted the concept of *Thing* (*Type*) from KCPM and adapted its properties to the needs of modeling for ambient assistance [15]. Things are there for modeling concrete and abstract objects or people (*Person*), which exist in the real world. For example within the context of a conference, the conference itself, papers, emails, journey, the registration system, a button of that system and others would be modeled as things. The optional attribute *Classification* allows distinguishing between things that are, from an object oriented point of view, classes and attributes. This distinction can be derived automatically as is shown in [7].

In order to relate (association *belongs-to*) the behavioral units to the right person and in order to cover interesting "static" properties, the target person (i.e. the person for whom a HCM is established) should be an instance of *Person*. Other instances are those persons that are involved in operations by the associations *calling, executing* and *participating*. As an example: When the flights for the conference journey are selected, the booking system requests payment information; in this case, the booking system is the calling thing, the user providing credit card data is the executing thing and the credit card number is a participating thing.

Location Things serve for modeling spatial aspects. If a person wants to book a flight but sits in the living room without a PC or Smartphone, HBMS has to help this person to get into a room with a PC or leads the person to a room, where she or he can pick up a Laptop or Smartphone and then return to the living room. The attributes of the concept Location Thing are inspired by [16]. Their values are persisted as vectors and will be collected using sensors and full-video analysis.

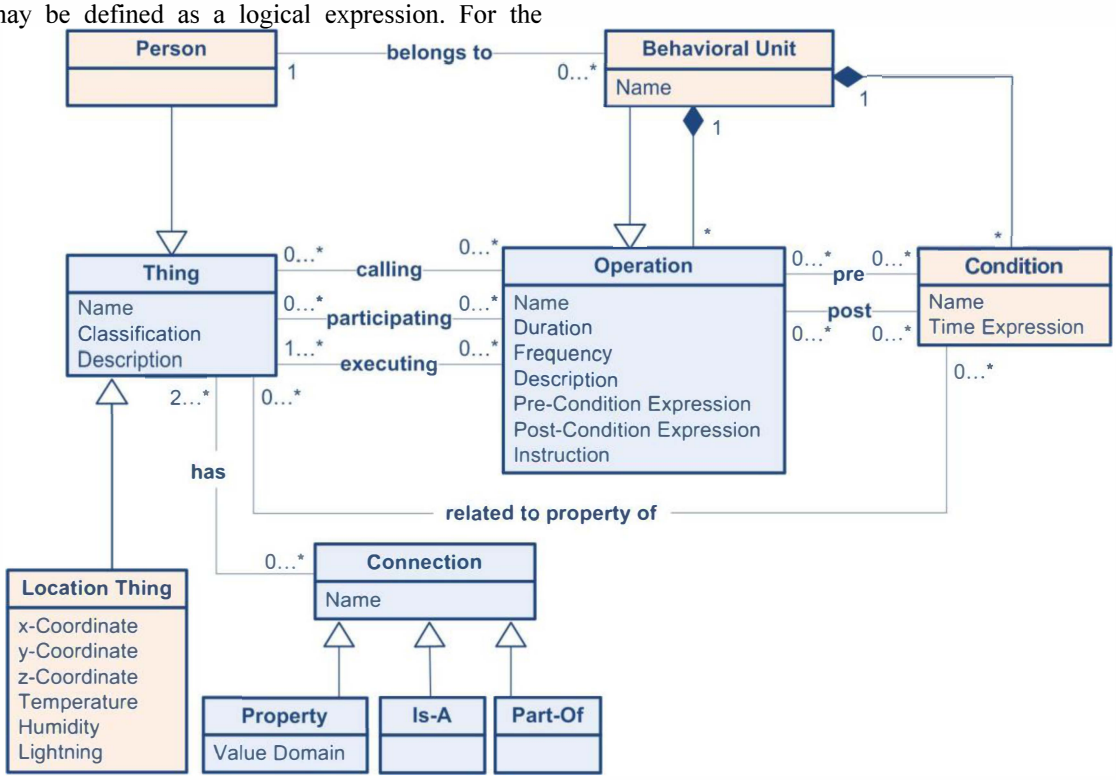

Fig. 1 Meta-Schema of the HCM (Definition of the HCM-L)

Things may be interrelated with other things in different ways. The concept of *Connection* covers all variants. For example, 'the shoes are in the entrance hall' is a classical association (relation); 'a shoe has a fastener (thing)' might be interpreted as an attribute relation (*Property*); the set of possible extensions of a thing is called *Value Domain* if this thing is targeted by a property. Examples for the thing fastener are velcro, lace, zip or snap fastener. Conditions refer to one or more properties and their concrete values.

Special types of connections are aggregation and decomposition (*Part-Of*) as well as specialization and generalization (*Is-A*).

The connection concept allows in particular to model specific topological aspects; as an example, consider some objects a person has in her/his hands (a mouse in the right hand, a cup of coffee in the left one); or the current hand positions, when this information is necessary for (supporting) a given activity. Together with the topological coordinates of the hand(s) accurate information about actions of a person may be gained for the HCM and used for hints when supporting that action.

C. Time

Operations can be executed at a specific time, e.g., you read the newspaper at 7 o'clock in the morning, drink your coffee every day at 9 o'clock in the morning, have to take your medicine at 10 o'clock and call your niece in Australia at 11 o'clock to reach her there at 7 o'clock pm local time after her working day. The condition attribute *Time Expression* covers these aspects.

Another point of interest is the time needed for executing an operation; e.g., the mobile TAN of an E-Banking application might be valid only for 5 minutes; it takes 3 minutes to heat up the water in the coffee machine after switching it on; and free WIFI access is available for one hour in a hotel. The attribute *Duration* of *Operation* covers these aspects.

Graphical Representation

The best choice for model presentation depends on the particular application (e.g. domain, validation, explanation) and situation. Thus, a modeling language like HCM-L must provide various representation concepts. From KCPM we adopted the glossary based (tabular) representation which mainly supports evaluation and validation purposes. In addition to that, a graphical notation supports explanation and visualization. Fig. 2 and 3 give an overview of the graphical symbols for HCM-L concepts as defined in the meta-schema shown in Fig. 1.

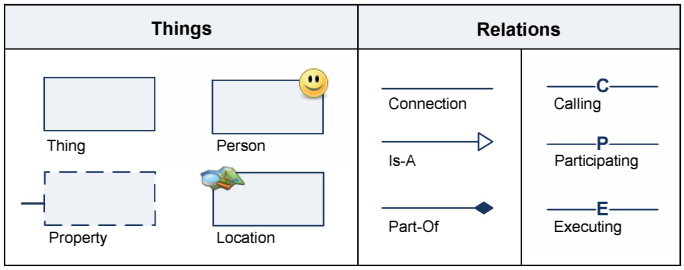

Fig. 2 HCM-L, the main symbols for modeling of static aspects

To model human behavior, we need symbols for modeling of dynamics. Fig. 3 shows the main symbols of HCM-L. *Operations* can have a *Pre-Condition Expression* and/or *Post-Condition Expression*. The symbol for an "operation with suboperation" reflects the case, where a behavioral unit again is seen as an operation (by generalization). This allows for hierarchical structures.

We have introduced symbols for *Conditions* and special variations of them for the *Start* and *End-Conditions*. The arrow shows the *Flow* between Operations and Conditions.

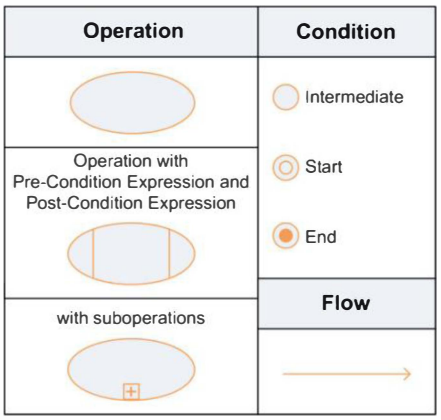

Fig. 3 HCM-L, the main symbols for modeling of dynamics

Integration

As has been explained before, behavioral units model a particular activity as an aggregate of operations with causal and/or temporal dependencies. The most simple cases are a single operation with some pre- and post-conditions and straight sequences of operations. However, human behavior varies over time and the particular situation, i.e. a person might carry out the same activity in more or less different ways. These variants have to be considered and modeled in order to empower the HBMS system to provide appropriate and flexible support. This leads to the necessity of (1) abstracting behavioral unit variants to more general and comprehensive representants of these variants as well as of (2) integrating different behavioral units to a coherent HCM. These tasks are solved by applying the integration framework discussed in [17]. It mainly focuses on identifying and solving concept matches and mismatches as well as conflicts and inconsistencies [18].

III. Pattern Analysis: Notions and Concepts

Pattern based analysis is a means to check a modeling language for completeness (w.r.t. to a given standard) and practicability. Since HCM-L is comprehensive in the sense that it covers structure, functionality and behavior we have to analyze its expressiveness regarding these three dimensions. Within this paper, however, we concentrate on behavior.

[2] describes a pattern to be "*a compact and rich in semantics representation of raw data*" and gives a two-level semi-formal definition as follows:

A *pattern type pt* is a quintuple *pt = (n, ss, ds, ms, f)* where:

(1) *n* is the *name* of the pattern type; (2) *ss (structure schema)* is a type in T[2] that defines the pattern space by describing the structure of the patterns instances of the pattern

[2] *T is a set of types including all base types together with all the types recursively defined by applying a type constructor to one or more other types*

type; (3) *ds (source schema)* is a type in T that defines the related raw data space by describing the dataset from which patterns are constructed; (4) *ms (measure schema)* is a type in T describing the measures which quantify the quality of the source data representation achieved by the pattern; (5) *f* is a *formula*, referring to attributes appearing in the source and in the structure schemas, that describes the relationship between the source space and the pattern space, thus carrying the semantics of the pattern.

A *pattern p instance of pt* is a quintuple *p = (pid, s, d, m, e)* where: (1) *pid (pattern identifier)* is a unique identifier for p; (2) *s (structure)* is a value for type ss; (3) *d (source)* is a dataset whose type conforms to type ds; (4) *m (measure)* is a value for type ms; (5) *e* is an expression denoting the region of the source space that is related to p.

Although the definition of pattern type seems blurred to some extent because of the verbal paraphrases it gives a good feel of the various aspects of patterns. For our purpose of pattern-based analysis of the HCM-L concepts for behavior modeling we concentrate on the structure schema of the relevant patterns.

The evaluation is done following [4] and [19] who applied the workflow pattern taxonomy given in [3] for analyzing UML and BPMN respectively; i.e. for each relevant pattern we present a HCM-L representation of it.

[3] distinguish the following four perspectives in process-aware information systems: (1) *control flow* (dependencies between various activities like concurrency, parallelism, choice and synchronization); (2) *data* (like passing of information, scoping of variables); (3) *resources* (like resource to task allocation, delegation) and (4) *exception handling* (causes of exceptions, actions). For the purpose of HCM-L the control flow perspective is the most important one. We therefore concentrate in the following on discussing HCM-L representations of the control flow patterns.

IV. Control Flow Pattern Analysis for HCM-L

For illustrating the analysis we mostly use examples from the scientific conference domain. Clearly, other domains like put on cloths, take medicine, use an E-Banking or E-Shopping application or use a complex device would be appropriate as well. [1] propose the following types of control flow patterns: (A) Basic Control Flow, (B) Advanced Branching and Synchronization, (C) Multiple Instance, (D) State-based, (E) Cancellation and Force Completion, (F) Iteration, (G) Termination and (H) Trigger. They will be discussed in that order.

A. Basic Control Flow Patterns

The Basic Control Flow Patterns (see figure 4) capture elementary aspects of a control flow:

Sequence (WCP1): serial routing of a process; a task can be executed only after completion of its predecessor task. As an example think of a login process with the subsequent operations 'fill in username' and 'fill in password'. The operations are connected by common conditions and flow arrows between them.

The *Parallel Split* (WCP2) forks a branch into two or more parallel ones which are executed concurrently. In HCM-L this can be modelled by an AND-headed post-condition expression of the preceding operation. E.g., we may book a hotel and a flight concurrently after having received the paper acceptance notification for a conference.

Synchronization or synonymously *AND-join* (WCP3) merges two or more branches to one so that the subsequent operation is executed only if or when all input branches have been successfully passed through. HCM-L provides this pattern by an AND-headed pre-condition expression of the subsequent (joined) operation. Example: If a hotel and a flight are booked, we need to check the visa regulations of the destination country.

The *Exclusive Choice* or synonymously *XOR-Split* (WCP4) selects exactly one of the branches of a split. In HCM-L we express this pattern by an XOR-headed post-condition expression of the preceding operation. E.g., after having registered for a conference in Vienna, we either can book a flight to Vienna or a train seat from our city to Vienna.

The *Simple Merge* or synonymously *XOR-join* (WCP5) reduces multiple branches to one such that control is passed if it comes from one unique branch. In HCM-L this pattern is reproduced by an XOR-headed pre-condition expression of the subsequent (joined) operation. E.g., after booking a flight or a train ticket to Vienna, we buy a single ticket for the Vienna Public Transport, which we will need anyway.

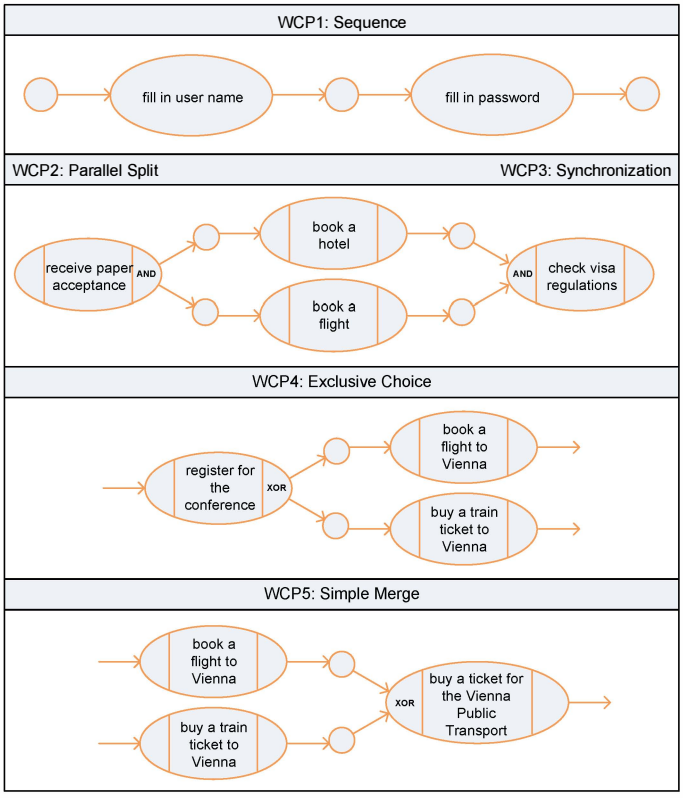

Fig. 4 Basic Control Flow Patterns in HCM-L

B. Advanced Branching and Synchronization Patterns

These patterns characterize a collection of more complex branching and merging situations: *Multi-Choice, Synchronizing Merge, Multi-Merge and Discriminator.*

The *Multi-Choice* (WCP6) or synonymously *OR-split* forks the process flow such that one or more subsequent branches may be performed concurrently. In HCM-L this is expressed by an OR-headed post-condition expression of the preceding operation. E.g., within a booking process we have several choices for hotel selection (see Fig. 5).

Synchronizing Merge is divided into three different Patterns: *Structured Synchronizing Merge* (WCP7), the *Local Synchronizing Merge* (WCP37) and the *General Synchronizing Merge* (WCP38). Until now, we recognized no need for the

two latter ones. The *Structured Synchronizing Merge* is always related to a Multi-Choice construct earlier in the process. It waits with the next operation until all incoming branches have been performed. Whereas UML and BPMN seem to have no sufficient solutions for this ([4], [19]), the HCM-L concept of *Synchronized OR (SOR),* used in the pre-condition expression of the joined operation, does the job. SOR always relates to the actual instance of the workflow. For example, the operation 'order the resulting hotel list by the user ranking, descending' (see Fig. 5) waits until hotels offering a safe AND those with free W-Lan have been chosen, if both pre-ceding operations had been started.

The *Multi-Merge* (WCP8) joins two or more branches to one. Each incoming branch results in an execution of the joined operation. HCM-L realizes this by just a simple operation, i.e. without a pre-condition expression, following two or more branches. An example can be seen in Fig. 5: No matter if we have booked a hotel or a flight, the process for paying via a credit card remains the same and must be executed every time the operation before is executed.

The *Discriminator* Pattern is dived into six WCPs: *Structured Discriminator* (WCP9), *Blocking Discriminator* (WCP28), *Cancelling Discriminator* (WCP29), *Structured Partial Join* (WCP30), *Blocking Partial Join* (WCP31) and *Cancelling Partial Join* (WCP32). They all have in common, that a subsequent operation may be executed when the first of a set of preceding operations has successfully been carried out.

The *Structured Discriminator* or synonymously *1-out-of-M join* waits for the first condition of "incoming branches" to be fulfilled, and then executes the joined operation; there are no more executions for the same process instance. In this case, there must be a Parallel Split construct somewhere earlier in the process which is related to this pattern. HCM-L offers for that pattern a SOR-headed pre-condition expression that "remembers" an execution of that operation within the current process instance. To explain it with the example in Fig. 5: The conference chair writes an email to some colleagues asking if they want to serve as a session chair for session 2. After the first positive reply he replies with further information and the papers for this session. All other positive replies do not evoke execution of this operation again.

The *Blocking Discriminator* is a variant of the Structured Discriminator; only the first fulfilment of a condition that is affected by an incoming branch enacts the execution of the subsequent operation. Further workflow instances are blocked at the Parallel Split until all other branches are processed. In HCM-L again a SOR-headed pre-condition expression of the joined operation will do the job which, in addition defines the blocking.

The *Cancelling Discriminator* is another variant of the Structured Discriminator; again only the first fulfilment of a condition that is affected by an incoming branch enacts the execution of the subsequent operation. In addition to that, the execution of all other operations on incoming branches is cancelled and reset. Since in the HBMS environment cancelling is behaviour to be modelled and potentially supported, there are no specific HCM-L concepts for that variant.

The *Structured Partial Join*, the *Blocking Partial Join* and the *Cancelling Partial Join* have the same functionalities like the *Structured Discriminator*, *Blocking Discriminator* and *Cancelling Discriminator* but for two and more branches (n-out of-m-join). HCM-L realizes these functionalities also by a SOR-headed pre-condition expressions and the definition that the joined operation is executed for each of the selected n branches.

The *Generalized AND-Join* (WCP33) which joins activities of different process instances is not needed for modeling the behavior of only one person.

C. Multiple Instance Patterns

Multiple instance workflow patterns concern situations where the same activity is executed concurrently, i.e. this activity has more than one active "running" instance [19]. As HBMS supports one acting person at the time multiple activity instances will not occur. Consequently, we can skip the related WCPs.

D. State Based Patterns

These patterns describe situations in which the current state of an activity influences the choice of flow alternatives.

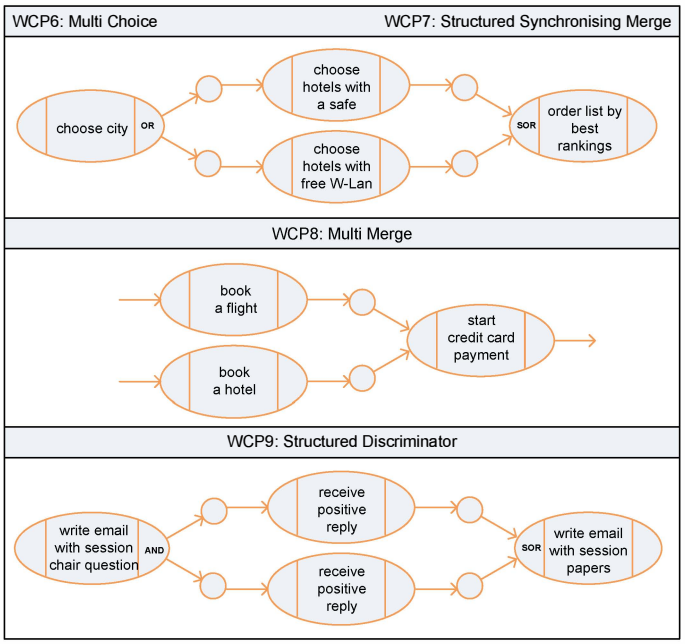

Fig. 5 Examples for the Advanced Branching and Synchronization Patterns

The *Deferred Choice* (WCP16) has a point, where one of several subsequent branches is chosen based on an interaction with the user. This pattern is covered by HCM-L similar to the Exclusive Choice pattern: there is a XOR-headed pre-condition expression of an operation that, in addition defines the source of the information necessary for the decision. The pattern is very important for support cases since in many everyday life situations the user decides on the choice of the particular actions and sequence out of several equitable ones; e.g., dressing for going out has many such actions like putting on shoes, putting on a hat, hanging a scarf around the neck etc. The selection of the actions and their sequence depends on season, weather, the user's mood and other reasons.

The *Interleaved Routing* (WCP40) has a section, in which operations can be executed in any order but no two at the same time. When all these operations are executed, the process can proceed. This is covered in HCM-L using the patterns Parallel Split and Synchronization (see Fig. 4); the mutual exclusion of the operations can be achieved by setting the "is Exclusive" attribute of these operations which is provided by HCM-L.

The *Interleaved Parallel Routing* (WCP17) is a variant of Interleaved Routing in the way such that there is a partial ordering of the operations: These can be executed in any order that complies with the partial ordering. Again no two operations are executed at the same time. In HCM-L the pattern is expressed equally to Interleaved Routing by extending the post-condition expression by the information about the partial ordering.

The *Milestone* pattern (WCP18) pertains to situations in which an operation may only be executed if a defined condition in another parallel branch holds. Using HCM-L this can be modeled by an AND-headed post-condition expression of the operation affecting the defined condition and an AND-headed pre-condition expression of the dependent operation. Fig. 6 depicts that case: there are two processes, namely (1) communicating about the purchase of a car with a car seller and (2) communication with a bank about contracting a leasing agreement. The salesman can only be assured about the purchase if there is a leasing commitment from the bank. This latter condition is the "Milestone".

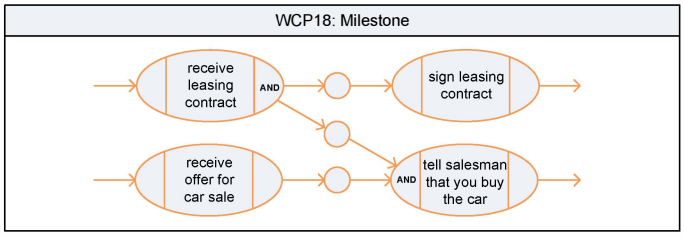

Fig. 6 Example for the Milestone pattern

In a Critical section (WCP39) only operations in one of several critical sections of different behavioral units can be executed at the same time. For this purpose, the HCM-L concept behavioral unit has an attribute 'is Exclusive' like operation. As an example, inserting and removing contact lenses are candidates for critical sessions.

E. Cancellation and Force Completion Patterns

These patterns describe various forms of cancelling an activity. The *Cancel and Complete Multiple Instance Activity Patterns* (WCP26 and WCP27) are not relevant since in the HBMS setting each behavioral unit belongs only to one individual and is executed only once at a time.

In contrast to that the *Cancel Task* (WCP19), *Cancel Case* (WCP20), and *Cancel Region* (WCP25) patterns are relevant in principle when modeling human behavior. However, since a person can cancel every operation at every moment, we decided to treat cancellation behavior like regular behavior; therefore no additional modeling concepts are needed. To cancel a process may have different reasons: a conscious decision, e.g. because the person wants to do something different; a problem in completing the process in question, e.g. because the person does not know the next step, because of a technical or another problem. Think of a situation where something goes wrong with credit card processing so that a payment cannot be completed. This kind of cancellations reflects typical situations for initiating HBMS support.

F. Iteration Patterns

Iteration Patterns reflect repetitive behavior.

Structured Loops (WCP21) execute one or more operations repeatedly. In HCM-L this will be realized using a XOR-headed post-condition expression of the cycle's last operation from which one flow points to the cycle's pre-condition. Fig, 7 shows as an example a situation, in which a person might repeatedly butter a slice of bread as part of a "having breakfast" activity.

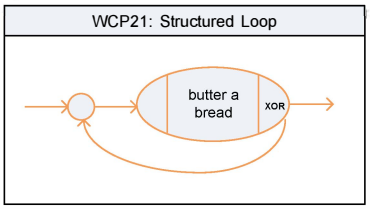

Fig. 7: Example for the Structured Loop

Arbitrary Cycles (WCP10) have more than one start and exit point. This can be emulated using HCM-L analogously to structured loops.

The *Recursion* (WCP22) pattern is not relevant for modeling human behavior.

G. Termination Patterns

In contrast to the cancellation patterns where we expect a process to be rolled back, Termination patterns concern situations in which an activity is simply abandoned or came to its natural end.

The *Implicit Termination* (WCP11) corresponds to an irregular end of the activity in question so that the activity's goal is not reached. This is a typical situation for initiating support by the HBMS system, that is in this case even more important than in the case of conscious cancellation since the termination might leave things in undesired states.

The *Explicit Termination* (WCP43) corresponds to a regular end of an activity where the related goal is achieved (see Fig. 8). HCM-L covers this with the end condition. As an example, the behavioral unit for booking a flight ends after the last operation, when the person has received the flight ticket by email.

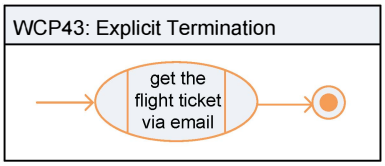

Fig. 8: Example for the Explicit Termination

H. Trigger Patterns

Trigger Patterns depend on external signals to set a pre-condition for an operation.

A *Transient Trigger* (WCP23) sets the respective pre-condition only for a certain period of time. If the related operation is not executed within that time, the condition is not more valid until reset by a new trigger signal. HCM-L provides for this purpose the attribute Time Expression in Condition (see Fig. 1) which allows formulating concrete points of time or durations of validity. As an example, an anonymous phone call can only be answered as long as it is ringing.

In contrast to that the effect of a *Persistent Trigger* (WCP24) is not volatile but persistent until the related operation is executed. In HCM-L this is modeled using a condition that has an in-flow from an external operation.

V. Future Work

We presented a pattern-based and control flow focused analysis of the HCM-L concepts for human behavior modeling.

It was shown, that regarding the workflow pattern taxonomy given in [2] all relevant situations are covered by HCM-L. However, further effort has to be spent into the analysis of the data, resource and exception handling perspectives (see section III) as well as of the HCM-L modeling concepts for static aspects.

Further research is also required concerning the formal specification of the language for time and condition expressions. This will be done in the next future.

REFERENCES

[1] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst and N. Mulyar. *Workflow Control-Flow Patterns: A Revised View*. BPM Center Report BPM-06-22, BPMcenter.org, 2006.

[2] S. Rizzi: *UML-based Conceptual Modeling of Pattern-Bases*, In Proceedings of the Intl. Workshop on Pattern Representation and Management (PaRMa 2004), Heraklion, Hellas, 2004.

[3] (2012) Workflow Patterns. [Online]. Available: http://www.workflowpatterns.com

[4] P. Wohed et al.: *Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams*. In Proc. Conceptual Modeling (ER 2005), Springer Berlin / Heidelberg, p. 63-78, 2005.

[5] J. Michael, A. Grießer, T. Strobl and H.C. Mayr: *Cognitive Modeling and Support for Ambient Assistance*. In: Proc. of the International United Information Systems Conference, UNISCON 2012, Yalta, Ukraine, Springer, 2012. [in press]

[6] C. Kop and H.C. Mayr: *Conceptual Predesign – Bridging the Gap between Requirements and Conceptual Design*. In Proceedings of the 3rd International Conference on Requirements Engineering. Colorado Springs Colorado, April 6-10, 1998.

[7] H.C. Mayr and C. Kop: *A User Centered Approach to Requirements Modeling*. In: M. Glinz et al. (Eds.): Modellierung 2002. Modellierung in der Praxis - Modellierung für die Praxis. Bonn: Köllen Verlag, Lecture Notes in Informatics (LNI) P-12, pp. 75-86, 2002.

[8] A.N. Leont'ev: *Activity, Consciousness, and Personality,* Englewood Cliffs, NJ, Prentice-Hall, 1978.

[9] A. Kofod-Petersen and M. Mikalsen: *Context: Representation and Reasoning,* Special issue of the Revue d'Intelligence Artificielle on "Applying Context-Management", 2005.

[10] A. Aamodt and E. Plaza: *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*, AI Communications, Vol. 7, pp. 39-59, 1994.

[11] W. Hesse and H.C. Mayr: *Modellierung in der Softwaretechnik: eine Bestandsaufnahme*. Informatik-Spektrum 31(5), p. 377-393, 2008.

[12] Th. Allweyer: *BPMN 2.0 Introduction to the Standard for Business Process Modeling.* BoD – Books on Demand, 2009.

[13] S. Katz: *Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living.* Journal of the American Geriatrics Society, Vol. 31(12), S. 721-727, 1983.

[14] M.P. Lawton and E.M. Brody: *Assessment of older people: Self-maintaining and instrumental activities of daily living*. Gerontologist 9, S.179-186, 1969.

[15] J. Michael and H.C. Mayr: *Benutzerzentrierte Modellierung für Ambient Assistance*, 2012, unpublished.

[16] F. Zhou et al.: *A Case-Driven Ambient Intelligence System for Elderly in-Home Assistance Applications*. Institute of Electrical and Electronics Engineers, New-York, 2011.

[17] J. Vöhringer and H.C. Mayr: *Integration of schemas on the pre-design level using the KCPM-approach*. In: A.G. Nilsson, R. Gustas, W.G. Wojtkowski, W. Wojtkowski, S. Wrycza, J. Zupancic (Hrsg.): Advances in Information Systems Development: Bridg-ing the Gap between Academia & Industry. Heidelberg: Springer Verlag, 2006.

[18] P. Bellström and J. Vöhringer: *Towards the Automation of Modeling Language Independent Schema Integration*. Proceedings of the 2009 International Conference on Information, Process, and Knowledge Management, IEEE Computer Society, 2009, pp. 110-115.

[19] P. Wohed et al.: *Pattern-based Analysis of BPMN – an extensive evaluation of the Controlflow, the Data and the Resource Perspectives*. Business Process Management, Lecture Notes in Computer Science, 2006, Volume pp 161-176.