Editors:
Tim Hussein, University of Duisburg-Essen
Stephan Lukosch, Delft University of Technology
Heiko Paulheim, SAP Research
Jürgen Ziegler, University of Duisburg-Essen
Gaëlle Calvary, University of Grenoble

Proceedings of the 2st International Workshop on

Semantic Models for Adaptive Interactive Systems (SEMAIS 2011)

Co-located with the 2011 International Conference on
Intelligent User Interfaces (IUI)
Palo Alto, CA, USA

Automated planning for User Interface Composition

Yoann Gabillon   Mathieu Petit   Gaëlle Calvary   Humbert Fiorino

University of Grenoble, CNRS, LIG

385, avenue de la Bibliothèque, 38400, Saint-Martin d'Hères, France

{yoann.gabillon, mathieu.petit, gaelle.calvary, humbert.fiorino}@imag.fr

ABSTRACT

In ubiquitous computing, both the context of use and the users' needs may change dynamically with users' mobility and with the availability of interaction resources. In such changing environment, an interactive system must be dynamically composable according to the user need and to the current context of use. This article elicits the degrees of freedom User Interfaces (UI) composition faces to, and investigates automated planning to compose UIs without relying on a predefined task model. The composition process considers a set of ergonomic criterions, the current context of use, and the user need as inputs of a planning problem. The user need is specified by the end-user (e.g., get medical assistance). The system composes a UI in turn by assembling fragments of models along a planning process.

Categories and Subject Descriptors

H.5.2 [User Interfaces]: Ergonomics, Graphical user interfaces (GUI), Prototyping, User-centered design. D2.2 [Software Engineering]: Design Tools and Techniques, User-Interfaces.

General Terms

Design, Human factors, Algorithms.

Keywords

User Interfaces composition, Semantic models, Automated task planning, Context of use.

1. INTRODUCTION

Pushed forward by new information technologies, Weiser's vision of ubiquitous computing comes to reality [11]. His definition of ambient computing implies 1) a global knowledge of an information system context, and 2) adaptation processes to comply with a given context of use. The context of use is usually defined as a <user, platform, environment> triplet. Unpredictable contexts of use might affect users' interactive behaviors and task organization. Therefore, each User Interface (UI) design option from the task model to the final UI is highly contextual and might be decided at runtime. Therefore, most of the ubiquitous design frameworks consider variations of the context of use as inputs to select UI options (i.e., plastic design [9], automatic generation [6], mashups [1]). However, to the best of our knowledge, the user task variation is usually left out.

This article outlines an approach, based on automated planning, to support task as well as UI variations in an integrated framework for UI composition. In the following, section 2 exemplifies multi-level UI composition on a medical support case study. Section 3 elicits the degrees of freedom UI composition faces to. Section 4 introduces automated planning and highlights the UI composition process. Section 5 presents an integrative framework for UI composition by planning. The focus is set on the composition of models (*Model-based composer*) and code (*Code composer*). Section 6 summarizes our contributions and draws some perspectives.

2. RUNNING CASE STUDY

Victor is a New-York citizen on vacation in Philadelphia. After spending his day tasting the rich local food, Victor feels bloated at night and needs to find the doctor on duty. Using his PDA, he specifies his need in general terms: "I would like to get medical support".

According to Victor's need and to the available interaction resources and existing information, the system *abstracts* the goal, *plans* a task model, and *composes* one possible UI. The composition process is not fully autonomous: it requires additional information from Victor. The negotiation UIs (Figure 1) are composed by the system as well.

Given Victor's current location, the system asks Victor whether he prefers to return home or to find assistance in Philadelphia (Figure 1a). Victor chooses to consult a local doctor. The system therefore finds and provides him with possible local contact information: the nearest hospital or doctor on duty, a medical hot-line, or the firemen (Figure 1b).
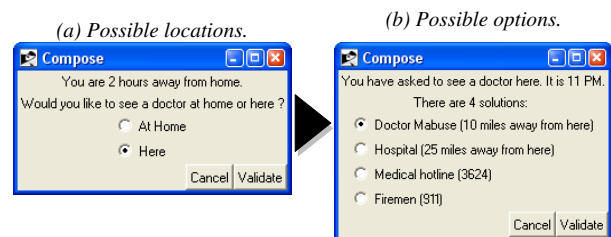

(a) Possible locations.   *(b) Possible options.*

Fig. 1. Automatically composed UI.

Victor selects the doctor on duty. The systems provides him with contact and location information. The UI layout matches the current user platform:

Smartphone. If Victor prefers to keep information at hand, a UI is generated for his Smartphone. With respect to the limited screen resolution, pieces of information are tabbed and no additional data is provided (Figure 2).

Fig. 2. The generated UIs for a Smartphone.

Desktop Wall. If a desktop wall is available, the system generates a single pane UI allowing to contact and/or to get route information to the doctor's office. Additional information about close services, like the nearest all-night chemist, is also provided (Figure 3).
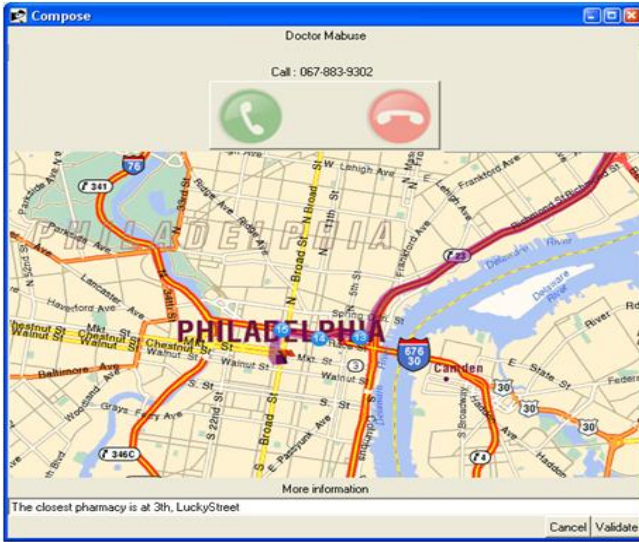

Fig. 3. The UI generated for a desktop wall display.

3. MODELS ARE KEY

This section goes back to model based design in Human Computer Interaction (HCI), and claims for keeping these models at runtime so that to support dynamic adaptation.

3.1 Model based design

UIs are modeled along several levels of abstraction. For example, the CAMELEON reference framework identifies four main levels of design decisions [2]. The *task model* (TM) describes how a given user task can be carried out; the *abstract UI* (AUI) delineates task-grouping structures (i.e., workspaces); the *concrete UI* (CUI) selects and layouts the interaction elements (i.e., interactors) into the workspaces; at last, the *final UI* (FUI) is about the code. Mappings relate these models to each other. For example, a task should be mapped to one workspace of the AUI at least.

In a dynamic context of use, any of these UI design decisions and their subsequent models and mappings might be updated at runtime to match the current context of use. As long as these adaptations satisfy the usability and utility properties, the UI is said to be *plastic* [9]. In Victor's case study, every design decision might be adapted in a plastic way. For example, the task "Find nearest chemist" may be removed from the task model. The AUI model associated to the Smartphone favors the "Call the office" subtask whilst the desktop wall version gives a simultaneous access to the two subtasks ("Call the office" and "Find route

information"). Variations at the CUI level are not exemplified in the case study. We could imagine a switch from a route display to a list of directions so that to fit with the Smartphone display. Such adaptations might be seen as a transformation between two graphs of models.

3.2 Graph of models to support adaptation

Earlier work defined principles for UI plasticity [8]. The authors structured the CAMELEON reference framework as a network of models and mappings (Figure 4), and claimed for keeping this graph alive at runtime so that to support adaptation.
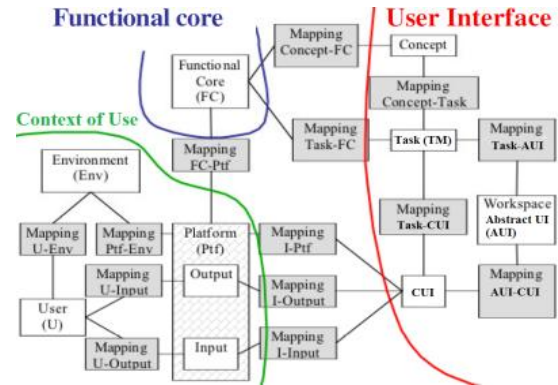

Fig. 4. Semantic graph of models of an interactive system [8].

The graph expresses and maintains multiple perspectives on a system. For example, a UI may include a task model, a concept model, an AUI model and a CUI model linked by mappings. In turn, the UI components are mapped onto items of the Functional Core, whereas the CUI interactors are mapped onto the input and output (I/O) devices of the platform. Although such a model provides a helpful organizational view on the elements and relationships involved when designing a plastic interactive software, the proposed mappings between the context of use and the other components hardly describe contextual choices inside each model (TM, CUI, AUI, etc.).

Demeure et.al. provide a complementary semantic graph of models to control UI plasticity within each design option level [4]. Their model allows UI designers to check out replaceable (i.e. functionally equivalent) units at run-time. For example, a given layout of interactors at the CUI level might be switched to another one depending on the desired ergonomic properties [7]. We propose to replace these hand-made choices by predicates dependent of the context of use, and manipulated by the system.

Figure 5 illustrates the design process along the models and mappings proposed in [8] and the replaceable options described in [4]. For example, at the task level (TM), two options exist for T2 depending on the context of use (Figure 5 b&c).

In Figure 5, within a level of abstraction, units relate to each other according to a *consumer-provider* relationship (Figure 5: $c \mapsto p$ link). For example, at the TM level, one of the options for the task T2 relies on the occurrence of a provider leaf option[1] for the task T3 (Figure 5a). Therefore, as T2 "consumes" T3, this option will be triggered if and only if T3 is satisfied. Depending on the current context of use, *consumer-provider* links behave like

---

[2] A *leaf option* has no relationship for neither providing nor reifying options.

"opened" or "closed" transistors. In a given $c \mapsto p$ relationship, the status of a transistor depends on the contextual requirements of the provider (p). For example, at the TM level in Figure 5, one of the task T2 options is possible only for experienced users (Figure 5d).
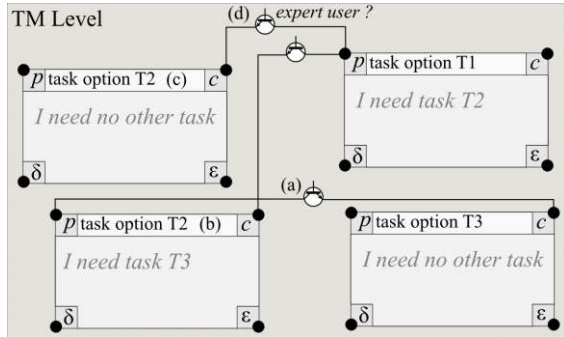

Fig. 5. Example of a TM options graph.

In UI design, mappings link together options of different levels of abstraction. For example, interactors from the CUI level are usually mapped onto workspaces of the AUI level. These mappings, presented in Figure 4, or the definitional links in [4] constitute *abstracting-reifying* relationships between the options of distinct CAMELEON levels of abstraction (Figure 6: $\delta \mapsto \varepsilon$ links).

Fig.6. Abstracting-reifying relationships between two design options at the TM and AUI levels of abstraction.

For example, the TM level presented in Figure 5 might be reified into several options of an AUI level (Figure 6). In Figure 6, a task option T1 is reified into a workspace layout "W3" of the AUI level. Like the $c \mapsto p$ relationship, $\delta \mapsto \varepsilon$ relationship between levels of abstraction makes sense in a given context of use only. For example, Figure 6 depicts a runtime configuration where the workspace layout W3 cannot reify the task T2 given the current context of use (Figure 6 a).

The relationships we propose ($\delta \mapsto \varepsilon$ and $c \mapsto p$) for modeling software can easily be explored automatically. The next section investigates automated planning.

4. UI COMPOSITION BY PLANNING

This section presents the core principles of planning and shows how this approach is valuable for UI composition.

4.1 Principles of automated planning

An automated planning algorithm derives a temporal sequence of *actions* into a *plan* to accomplish a given *goal* [5]. For example, in

the previous case study, the sequence {"Call the doctor"→"Find route information"} is a plan made of two actions. A Planning algorithm pipes syntactic processes to perform symbolic computations. Such logical reasoning is formally described by a finite-state machine where actions are transitions between possible *states of the world*. Actions are defined by sets of pre/post-conditions. Pre-conditions specify the run-time dependencies of an action while post-conditions are met after executing the action. For example, Victor's Smartphone should be connected *(pre-condition)* to display a location map *(action)*. When this action is executed, the map is eventually displayed *(post-condition)* on the Smartphone. An updated state of the world integrates these new post-conditions, therefore enabling further actions.

4.2 Automated planning for UI composition

A planning solver algorithm computes a transition graph between an initial state of the world and a final state corresponding to the system/user goal. Currently, such algorithms are mainly applied to service composition [10]. However, as illustrated in our case study, context-dependent UI composition and automated planning strongly relate. Thus, we propose to address UI composition by planning where:

- "Actions" are "User interfaces options". Existing components (e.g., the UI associated to the task "Call the office") are actions for the planner;
- The "State of the world" is made of the current "Context of use" and the "Ergonomic properties" to be satisfied. For example, the fact *"Victor owns a Smartphone"* is a predicate of the state of the world;
- The "selected plan" is the "composed UI". For example, the UI displayed on the Smartphone is a concretization of the plan {"Choose the city"→"Choose the doctor" →"Contact the doctor"→{"Call the office"→"Find the route information"→ "Find the nearest pharmacy"}} computed by the planner.

Even if several challenges still need to be worked out to bridge the gap between automated planning and UI composition, next section presents "Compose", a first framework for rapidly prototyping UIs by planning. Its use by end-users belongs to the future.

5. THE COMPOSE FRAMEWORK

Compose is a proof of concept of UI composition by planning. It has been built on top of several functional Java-coded components (Figure 7).
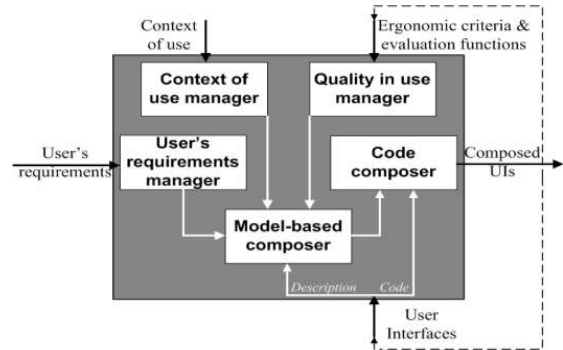

Fig. 7. Functional decomposition of Compose.

The *Context of use and quality in use managers* translate the required ergonomic criteria and the current context of use into

predicates. These assertions define the current state of the world. For example, the predicate *Has("User","Desktop Wall")* is true when Victor stands nearby a managed desktop wall.

The *User requirements manager* expresses a user need as a goal to be met. For example, Victor's need would be to "Get medical support".

The *Model-based composer* and the *code composer* are the core components of Compose. The model-based composer handles the planning process, whilst the code composer translates a resulting plan into a FUI. In the current prototype, planning is applied to the task level only. Once the TM level is composed, mappings are made with a generic purpose graphic toolkit called COMET [3]. COMETs are reusable context-aware widgets defined at the task level and reified along the CAMELEON reference framework. The next sections focuses on the core components of Compose.

5.1 Model-based composer

The model based composer takes actions as inputs and structures them into a plan. This planning process is twofold: at first, the user task modeling is composed by collating predefined subtasks (Figure 8(p1)); next, each task (i.e.: the planner actions) is mapped onto a UI (Figure 8(p2)). These selections bring out a composed UI (i.e., the selected plan) whose properties match the current state of the world. The resulting plan is a semantic description of the UI to be composed.
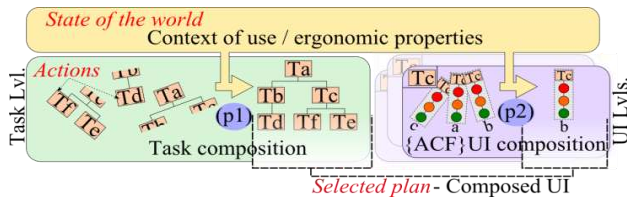

Fig. 8. Compose planner instantiation.

In Victors' case study, Compose waits for a user need specification (i.e. "Get medical support"). The composer tries to find a corresponding TM level entry point. The option "Get Medical Support" is selected. The planning algorithm then explores the semantic network of $c \mapsto p$ relationships between the task options of the TM level (Figure 9). For each uncovered task option, Compose checks whether it is possible or not to map the task onto a COMET and render the UI. These mappings are derived according to the current state of the world. For example, leaf task options like "Choose the city" or "Choose the doctor" might be mapped onto a UI as soon as Victor's platform is available whatever the characteristics of the platform are (in Figure 9: t1 & t2). Other task options like "Call the office" rely on carrier capabilities at the platform level (in Figure 9: t3). "Contacting the doctor" option distinguishes between several screen sizes and resolutions (Figure 9: t4 & t5). When a large screen is available, such a sub-task option involves tree leaf options (Figure 9: u1), while on a Smartphone display, solely two of them are displayed (Figure 9: u2).

Once all contextual pre-requisites of a provider option are met, the relationships to his consumers turn green and each of them might in turn be checked-out. After a provider/consumer relationship status has been specified, the state of the world is updated with the new facts the providing option concurs to establish. For example, when "Choose the city" pre-requisites are met, the composer knows for sure that Victor will be able to specify his searching

location and the fact "The location has been set" is added to the state of the world.

Figure 9 outlines the status of the $c \mapsto p$ relationship between the task options after Compose has explored and checked-out a state of the world wherein Victor interacts on a desktop wall display.
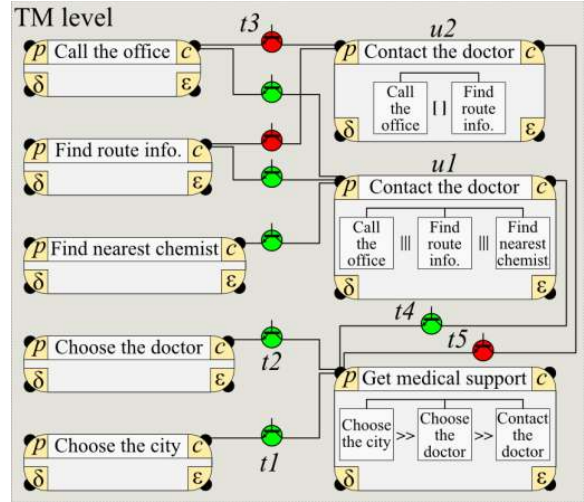

Fig. 9. Possible TM level planning when a desktop wall is available.

Such contextualized semantic UI model highlights the appropriate task factorization in a given context of use. When a green path of provider-consumer relationship is established from the provided objective to the leafs task options, a task tree has been found to achieve the user goal. In such case, the code composer is provided with the planned task tree. Subsequent mappings are made between tasks and COMETs to derive the final UI.

5.2 Code composer

The *Code composer* derives the UI code from the graph of models at the task level. At design time, the options of the task level have been statically associated to COMETS. Therefore, in Compose, each action of the plan is reified by a contextualized COMET. For example, the option "Get medical support" is mapped to a COMET laying out a sequence of frames on the desktop wall. The *Code Composer* brings these pieces of UI together in a unified layout. For instance, the desktop wall task tree provided by the model-based composed is mapped to the COMET presented in Figure 10. For example, the action "Get medical assistance" is mapped to a "COMET C7" laying out a sequence of frames on the desktop wall. These frames contain several sub-COMETs ("COMET {C3, C1, C4}") to map the task options "Choose the city", "Choose the doctor" and "Contact the doctor". In turn, the mapping "COMET C4", that reifies the task "Contact the doctor", contains several vertically aligned sub-COMETs. These sub-COMETs ("COMET {C2, C5, C6}") are mapped in the same way.
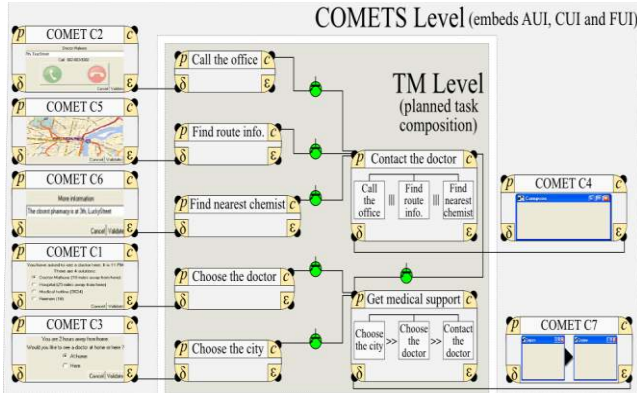
Fig. 10. The "Desktop Wall" planned task tree. Each task is reified by a pre-defined COMET.

6. CONCLUSION AND FUTURE WORK

This article outlines a work in progress to support opportunistic user needs. A UI is composed by selecting a path in a graph of models according to the current context of use and the ergonomic properties to be satisfied. UI composition is seen as a planning problem. So far, the focus has been set on the model-based composer whatever the *time* is: design time for the designer thus providing a rapid prototyping tool, or runtime for the end-user as an intelligent assistant.

Future works include improvements of planners to fully support UI composition. This means (1) generating trees (i.e., tasks structures) instead of sequences, (2) defining appropriate functional and implementational software architectures for general-purpose ubiquitous computing, (3) taking non functional properties into account (i.e., returning the best plan instead of the first one). Thus, beyond perspectives in HCI, this work has challenged planning for ubiquitous computing.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] Brodt, A., Nicklas, D., Sathish, S., and Mitschang, B. 2008. Context-aware mashups for mobile devices. In *WISE 2008: Web Information Systems Engineering*. Springer-Verlag, 280-291.

[2] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. 2003. A unifying reference framework for multi-target user interfaces. *Interacting with Comp. 15*, 3, 289-308.

[3] Demeure, A., Calvary, G., and Coninx. 2008. K. COMET(s), A Software Architecture Style and an Interactors Toolkit for Plastic User Interfaces. In *15th Int. Work. on Interactive Systems Design, Specification, and Verification*. Springer-Verlag, 2008, 225-237.

[4] Demeure, A., Calvary, G., Coutaz, J. and Vanderdonckt, J. 2006. The COMETS Inspector: towards run time plasticity control based on a semantic network. *In Proceedings of the 5th Int. Workshop on Task Models and Diagrams for User Interface Design: TAMODIA'06*, Springer LNCS 4385, Haselt, Belgium, 324-339.

[5] Nau, D., Ghallab, M., and Traverso. P. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[6] Paternò, F., Mancini, C., and Meniconi, S. 1997. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, Chapman & Hall, Ltd., 362-369.

[7] Scapin, D.L. and Bastien, J.M.C. 1997. Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behaviour & Information Technology.* Colchester, ROYAUME-UNI: Taylor & Franci *16*, 4 (1997), 220-231.

[8] Sottet, J-S., Ganneau, V., Calvary, G., Coutaz, J., Demeure, A., Favre, J-M. and Demumieux, 2007. R. Model-driven adaptation for plastic user interfaces. *In Proc. of the 11th IFIP TC.13 Int. Conf. on Human-Computer Interaction : INTERACT'07*, Springer LNCS 4662, Rio de Janeiro, Brazil, 397-410.

[9] Thevenin, D. and Coutaz, J. 1999. Plasticity of user interfaces: Framework and research agenda. *Human-computer Interaction, INTERACT'99: IFIP TC. 13 , 30th August-3rd September 1999*, IOS Press, 110.

[10] Traverso, P. and Pistore, M. 2004. Automated Composition of Semantic Web Services into Executable Processes. *Proceedings of ISWC*, *LNCS*, 380-394.

[11] Weiser, M. 1991. The computer for the 21st century. *Special Issue on Communications, Computers, and Networks 272*, 3, 78-89.

Generating High-Level Interaction Models
out of Ontologies

Dominik Ertl, Hermann Kaindl,
Edin Arnautovic, Jürgen Falb,
and Roman Popp
Vienna University of Technology
Institute of Computer Technology
(ertl, kaindl, arnautovic, falb,
popp)@ict.tuwien.ac.at

ABSTRACT
Generating user interfaces out of semantic models is still
an issue because of the semantic gap between ontologies
and user interfaces. We bridge this gap through semantic
model-driven development. More precisely, we show how
to automatically generate high-level interaction models (in
the form of communication models representing discourses)
out of (annotated) ontologies, using model-transformation
rules. From these discourse models, user interfaces can be
generated (semi-)automatically.

INTRODUCTION
The most important elements of any interactive system are
the information it contains and the user interface through
which this system communicates with its users. The infor-
mation may be represented with (formal) semantic models
(e.g., based on ontologies), and the user interface is typi-
cally created manually on top of such models. This requires
a lot of effort, especially if these models are modified and
the user interface has to be adapted manually.

In a specific category of interactive systems, such as product
recommendation systems, reservation systems or shopping
applications, the underlying (semantic) model may strongly
influence the behavior of the systems and, therefore, also the
interactions to be implemented through the user interfaces.
For this category of interactive systems, we address the se-
mantic gap between underlying ontologies and user inter-
faces. We make use of our *discourse models* [1, 3] for bridg-
ing this gap. In this course, a discourse model and a domain-
of-discourse model together serve as a high-level interaction
model and, as such, as a kind of "intermediate language" be-
tween the ontology and the user interface. In addition, such a
model can even be used for the (semi-)automatic generation
of a user interface [3, 10].

The remainder of this paper is organized as in the follow-
ing manner. First we give a brief background on our previ-
ous work relevant for this paper, and compare it with some
of the related work in the field. Then we present our ap-
proach for generating interaction models out of (annotated)
ontologies. This approach contains two parts: the generation
of *discourse* models representing the flow of communica-
tion between the user and the computer, and the generation
of *domain-of-discourse* models representing what they "talk
about". Finally we conclude and provide an outlook of our
future work in this direction.

BACKGROUND AND RELATED WORK
Our previous work focused on manual *modeling* of interac-
tion designs [1], where even end users created interaction
designs in the form of discourse models using the graphi-
cal editor developed for this purpose. These discourse mod-
els are based on several theories of human communication
[2]. The key parts of our discourse models are *Commu-
nicative Acts* as derived from speech acts [11], *Adjacency
Pairs* adopted from Conversation Analysis [5], and *RST rela-
tions* inherited from Rhetorical Structure Theory (RST) [6].
Communicative Acts are semi-structured messages carrying
the intention (e.g., asking a question or issuing a request)
and represent basic units of language communication. Ad-
jacency Pairs are sequences of talk-turns that are specific to
human (oral) communication, e.g., a question should have a
related answer. RST relations specify relationships among
text portions and associated constraints and effects, and are
organized in a tree structure. In our work, we use RST for
linking Adjacency Pairs of Communicative Acts and further
structures made up of RST relations. We have also included
procedural constructs, to provide means to express a partic-
ular order during discourse execution, to specify repetitions
or conditional execution of different discourse parts. Since
such discourses cast the communication between a human
and the computer on a high level, abstracting from technical
details, they may even be created without any programming
knowledge and experience.

Instead of our discourse models, ConcurTaskTrees from Pa-
terno et al. [7] may be used for bridging the semantic gap be-
tween ontologies and user interfaces. ConcurTaskTrees fa-
cilitate modeling *tasks*, that are being transformed into a user
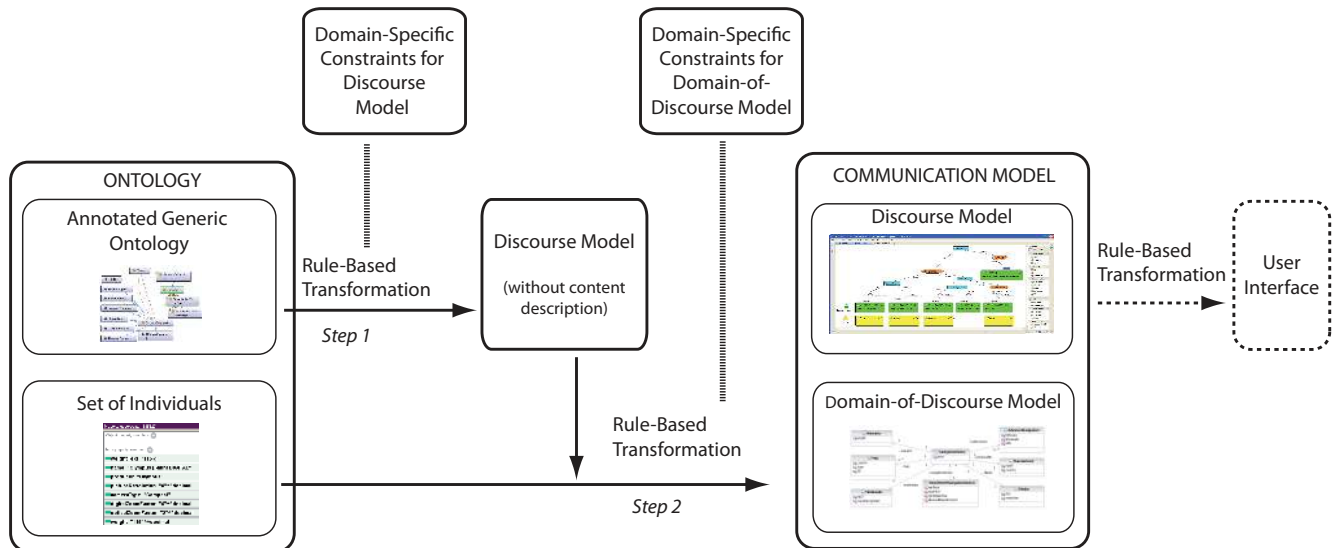interface. Our discourse models focus more on the commu-

Figure 1. Transformation from ontology to communication model.

nication, and they can be used for machine-machine communication as well [9]. According to our best knowledge, this has not been done with ConcurTaskTrees. We are also not aware of any approach for generating ConcurTaskTrees out of ontologies.

UsiXML [4] is an XML-based specification language for user interface design. It allows describing a user interface at different levels of abstraction, from high-level task models to the concrete code of a user interface. So, it provides an alternative approach to ConcurTaskTrees. Also for UsiXML, we are not aware of any approach for generating UsiXML models out of ontologies.

Paulheim and Probst [8] present a survey about ontology-enhanced user interfaces. They point out that ontologies can be used to improve interaction possibilities, and our approach addresses such a possibility.

FROM ONTOLOGIES TO INTERACTION MODELS

Now let us present our approach to automatically transforming an ontology to a high-level interaction model in the form of a specific communication model by using model transformations. We focus on a small part of an ontology and its corresponding transformations to generate the interaction model of a Product Advisor for digital cameras as a running example. The Product Advisor is designed to ask questions about desired properties of a digital camera to be bought.

Overall Transformation Approach

In Figure 1, we provide an overview of the transformation process, which consists of two steps for generating a communication model from an annotated ontology. We start from such an ontology represented in OWL[1] (illustrated in the left part of Figure 1). While the ontology per se con-

tains the knowledge of the given domain, the annotations contain *meta-knowledge*, e.g., the priority of a given piece of knowledge with respect to the Product Advisor to be implemented. The result is a communication model consisting of a discourse model and a domain-of-discourse model.

We use a GoodRelation[2] ontology for digital cameras as a basis. In Figure 2, we depict selected parts from the Digicam GoodRelation ontology. The top concept *Thing* is specialized by the concepts *ProductOrService* and *DomainSegment*. ProductOrService is further specialized by the *Digicam* concept. DomainSegment groups together properties of a ProductOrService that have a semantic relation with each other.

Our ontology contains additional annotations that describe characteristics of certain datatype and object properties with respect to the intended Product Advisor. For example, the annotation *priority* specifies how important for the Product Advisor a specific object or datatype property is compared to other properties. These priorities are a distinguishing feature when the transformation process applies the transformation rules. The priority is an integer value between 0 (low priority) and 100 (high priority). The priorities allow the transformation process to decide which datatype and object properties are of interest for the discourse and the domain-of-discourse.

In the first step, a set of model-transformation rules matches parts of the ontology (including its individuals) and transforms them automatically into corresponding parts of a discourse model (see the middle part of Figure 1). These transformations are subject to domain-specific constraints explained in detail below. The discourse model generated in this step represents only the generic communication flow

---

[1]Last visited on December 10, 2010: `http://www.w3.org/TR/owl2-overview/`

[2]Last visited on December 10, 2010: `http://www.heppnetz.de/projects/goodrelations/`
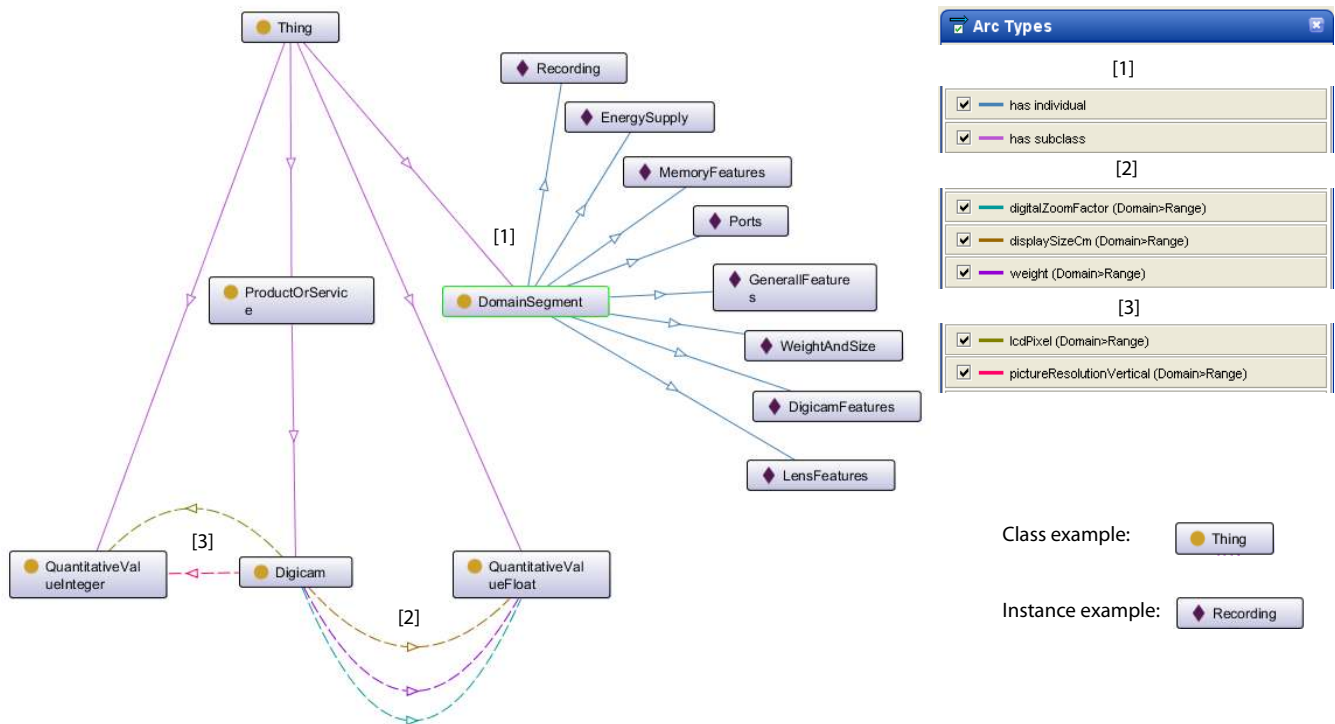
Figure 2. Selected parts from Digicam ontology.

and is incomplete since the *content* of its communicative acts does not (yet) refer to the content of the communication (the domain-of-discourse model).

In the second step, our model-transformation approach transforms the individuals of the ontology and their concrete datatypes and object property values into a domain-of-discourse model. Here we apply domain-specific constraints for a domain-of-discourse model. In effect, this step defines the content of the communicative acts so that the discourse model refers to the domain-of-discourse model. The contents of communicative acts in the case of a Product Advisor are concrete question and answer texts that link to elements of the domain-of-discourse model.

From Ontologies to Discourse Models
Our transformation approach applies several rules to create a discourse and a domain-of-discourse model out of the ontology. Each rule application can be constrained by domain-specific constraints, that are externally configured. We have a logical rule chain (by using Operational Query/View/-Transformation[3] (QVT)) defining the application order of the rules. In principle, a rule that is applied later in the transformation process can influence the outcome of a rule that is applied sooner. In the following, however, we describe two independent rules applied in the transformation process, the *DomainSegmentClusterRule* and the *SingleQuestionRule*.

The first rule explained as an example is the DomainSeg-

mentClusterRule illustrated in Figure 3. It matches the concept *DomainSegment* in the ontology, which has several individuals. For example, the digital camera ontology has the domain segments EnergySupply, LensFeatures, Ports, etc. All datatype and object properties in the ontology that are related to a DomainSegment via the object property *belongsToDomainSegment* are of interest for our transformation process.

The rule DomainSegmentClusterRule creates a *cluster* of questions for all object and datatype properties that belong to the same domain segment. A cluster groups questions that hold a semantic relation (e.g., the ports USB and FireWire belong to the DomainSegment *Ports*). Such a definition of a cluster results in a *Joint* RST relation of a discourse (see the right part of Figure 3). The datatype and object properties are transformed into question/answer pairs that are branches of the Joint relation. In addition to the rule presented above, the following domain-specific constraint applies: Each property needs a *minimum priority* value (e.g., 20) to be included in a cluster. The minimum priority value is configured a priori in the domain-specific constraints.

After the DomainSegmentClusterRule has been applied, all properties with a minimum priority are grouped in the different clusters. For example, USB belongs to the segment *Ports*. Now a rule applies that combines all Boolean properties (like USB) of one domain segment into one question, for optimizing the interaction with the Product Advisor. The left part of Figure 4 shows the datatype property *USB*, which represents the USB port of a digital camera having the priority 85. For this USB property, the *SingleQuestionRule* takes
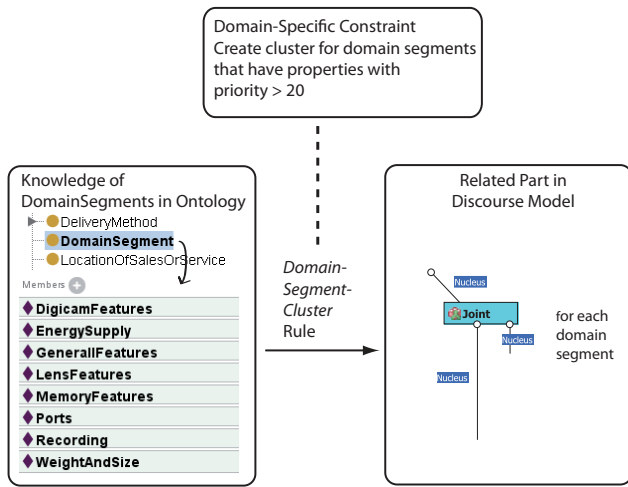
---

[3]Last visited on December 10, 2010: `http://wiki.eclipse.org/M2M/Operational_QVT_Language_%28QVTO%29`

Figure 3. Transformation with DomainSegmentCluster Rule.

Figure 4. Transformation with SingleQuestion Rule.

effect now. Due to its high priority value (and importance), USB becomes a single ClosedQuestion-Answer pair again. This adjacency pair, shown in the right part of Figure 4, becomes also part of the generated discourse model.

As shown with these example rules and their applications, for each selected property a corresponding question is being generated for the Product Advisor, since this property is considered important for the selection of a camera. While the ontology specifies what exists in the domain, the process being implemented in the Product Advisor contains related questions. This semantic gap is bridged by our approach in the context of the given application.

We show an excerpt of a yet incomplete discourse model in Figure 5, that is the result of the first transformation step depicted in Figure 1. The contents of the communicative acts are URIs that refer to datatype or object properties in the ontology. A *Joint* relation combines one Adjacency Pair and the Background relation connecting two more Adjacency Pairs. In this example, these properties have a high enough priority, so that they have to be grouped together in a special cluster at the beginning of the recommendation process of the Product Advisor. The first question gathers information on the price range, defining the minimum and maximum price that the user is potentially willing to pay. The second question elicits the interest for a USB port on the digital camera. This Boolean question is modeled as a closed question. Moreover, there is an RST relation *Background* intended to optionally inform the human user on additional details about the subject matter, e.g., more information on USB.

From Ontology to Domain-of-Discourse Model
The second step in the transformation from ontologies to our communication models is to generate the *domain-of-discourse model*. This model represents the content of the communication, more precisely the content of the communicative acts within our discourse models. For example, the digital camera's property *hasCurrencyValue* (representing the price of the camera) is the content of the question
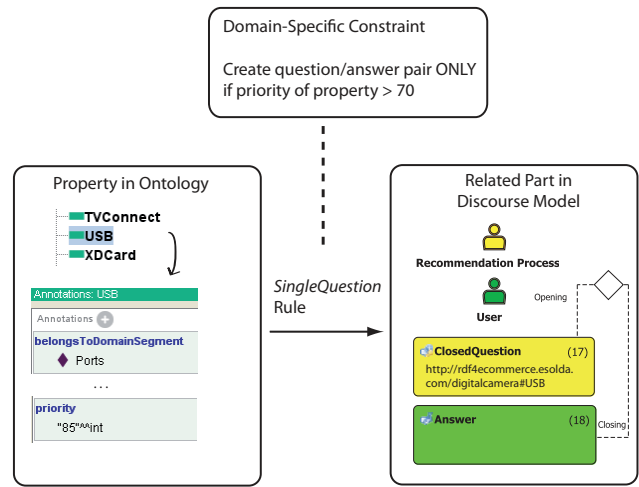
where the Product Advisor asks the user about his or her preferences (e.g., price range) regarding the camera price (shown at the top of Figure 5). The Product Advisor should only ask for relevant product properties and their values. For example, the prices of all cameras should be within the price range offered for selection. So, the set of individuals of the products is used to generate the possible contents of the communicative acts, e.g., to determine their price range.

So, for the content of each question in the discourse model, a unique *datatype* representing the product property is generated in the *domain-of-discourse model*. Figure 6 shows a small excerpt of such a generated domain-of-discourse model. For product properties representing *numbers* (e.g., price), only the minimum and maximum values are relevant for the Product Advisor (e.g., to generate a slider in the final UI for selecting the preferred value between the minimum and maximum). These values are stored together with the generated datatype. The left part of Figure 6 shows the datatype of the price property realized by a *Float* number. The minimum and maximum values are displayed as an annotation in a *note* below the datatype. For product properties representing *Boolean* values, the concrete individuals do not have to be searched for possible values, of course. As an example of such a Boolean datatype, the USB datatype is shown in the middle of Figure 6. For all *other* properties, an *Enumeration datatype* is generated for storing all possible values. The right part of Figure 6 shows the Enumeration type generated for the *producer* datatype. The values of the enumeration are derived from the set of all camera producer individuals in the given ontology.

The applications of these transformation rules can be influenced by domain-specific constraints specific for the domain-of-discourse model. For example, if no values for a specific property exist in the set of individuals in the ontology or if all of them are same (e.g., if all cameras have a USB interface), then the content of the question would be empty, so that the whole question is deleted from the discourse model.
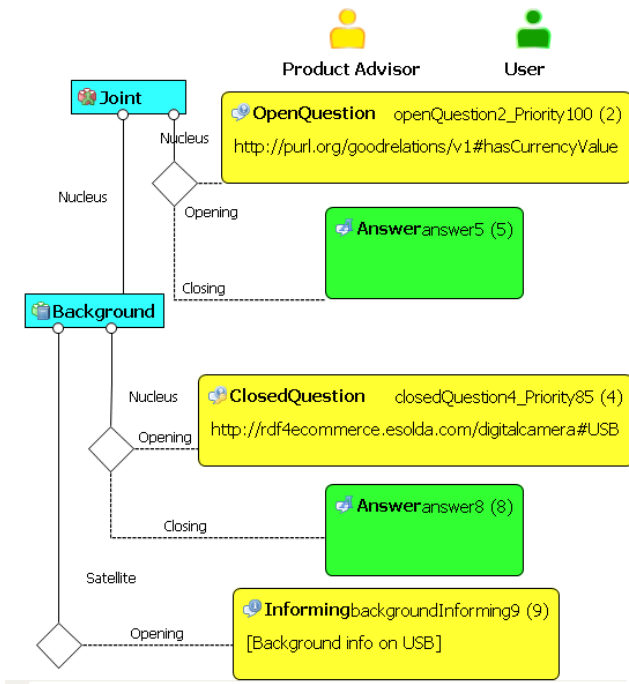
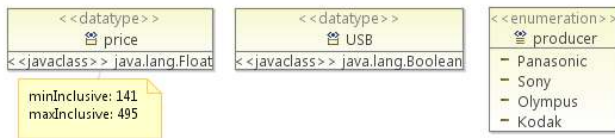**Figure 5. A cluster of questions with high priority.**

Figure 6. Excerpt of Domain-of-Discourse Model.

CONCLUSION

To ease and speed up the development of ontology-based interactive systems, the automatic generation of their user interfaces would be advantageous. However, due to different perspectives as well as technical and conceptual foci of ontologies used in such systems, the generation of user interfaces directly from ontologies would be hard. We use a high-level interaction model in the form of a communication model based on discourses as an intermediate language. In this paper, we explain the automatic generation of such models out of (annotated) ontologies, and taking application-specific constraints into account.

From such communication models, user interfaces can be generated (semi-)automatically, as we have previously shown already [3]. For small devices, even fully automatic generation leads to usable interfaces through special optimizations of the use of the constrained space [10].

Acknowledgment

REFERENCES

1. C. Bogdan, H. Kaindl, J. Falb, and R. Popp. Modeling of interaction design by end users through discourse modeling. In *Proceedings of the 2008 ACM International Conference on Intelligent User Interfaces (IUI 2008)*, Maspalomas, Gran Canaria, Spain, 2008. ACM Press: New York, NY.

2. J. Falb, H. Kaindl, H. Horacek, C. Bogdan, R. Popp, and E. Arnautovic. A discourse model for interaction design based on theories of human communication. In *Extended Abstracts on Human Factors in Computing Systems (CHI '06)*, pages 754–759. ACM Press: New York, NY, 2006.

3. J. Falb, S. Kavaldjian, R. Popp, D. Raneburger, E. Arnautovic, and H. Kaindl. Fully automatic user interface generation from discourse models. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI '09)*, pages 475–476. ACM Press: New York, NY, 2009.

4. D. Faure and J. Vanderdonckt. User interface extensible markup language. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*, pages 361–362. ACM Press: New York, NY, 2010.

5. P. Luff, D. Frohlich, and N. Gilbert. *Computers and Conversation*. Academic Press, London, UK, January 1990.

6. W. C. Mann and S. Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

7. F. Paterno, C. Mancini, and S. Meniconi. ConcurTaskTrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Sixth International Conference on Human-Computer Interaction*, pages 362–369, 1997.

8. H. Paulheim and F. Probst. Ontology-enhanced user interfaces: A survey. *Int. J. Semantic Web Inf. Syst.*, 6(2):36–59, 2010.

9. R. Popp. Defining communication in SOA based on discourse models. In *Proceeding of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications (OOPSLA '09)*, pages 829–830. ACM Press: New York, NY, 2009.

10. D. Raneburger, R. Popp, S. Kavaldjian, H. Kaindl, and J. Falb. Optimized GUI generation for small screens. In *LNCS Volume on Models in Software Engineering: Workshops and Symposia at MoDELS 2010*. Springer, 2011.

11. J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, England, 1969.

Adaptive presentation of itineraries in navigation systems by means of semantic models

Daniel Muenter & Tim Hussein
University of Duisburg-Essen
Lotharstr. 65, 47057 Duisburg, Germany
{daniel.muenter, tim.hussein}@uni-due.de

ABSTRACT
In this paper, we introduce a technique for adaptive presentation of itineraries in navigation systems based on semantic models. We enrich waypoints with semantic information and display only those waypoints to the driver that he is really interested in, hiding information that will most probably be distracting.

Author Keywords
Model-driven UI Generation, Navigation Support

ACM Classification Keywords
D.1.2 Software: Programming Techniques—*Automatic Programming*

INTRODUCTION
Navigation systems are widespread tools in automobiles. According to recent german studies [5], the percentage of pre-installed navigation systems increased from less than 6% to 18% within the last six years (in Germany). The percentage of mobile navigation systems even rose from 1% to almost 31% in the same period. With regard to usability [7] and traffic routing [3, 11], constant progress has been made during the last years. However, there is room for improvement in many ways.

Usually, the presentation of the itinerary is very detailed – even if the driver knows parts of the route very well. This is often distracting and annoying. Presentation techniques that take the users knowledge and driving behavior into account can improve the user experience considerably.

Present solutions aim at optimizing routes without taking the driver's personal knowledge, experience, and preferences into account and, thus, are not personalized. However, incorporation of personal information could improve presentation of routes significantly. On the one hand, instructions should be rather short and abstract, if the user knows the particular area, and, on the other hand, more detailed, while driving through unknown territory.

In this paper, we introduce a concept to enhance the presentation of the route by adapting it to the driver and his preferences and experience. For that purpose, we use semantically enriched models of the itineraries. In the end, the user should only see and hear *necessary* and helpful information instead of every single detail. Besides automated adaptation, the user has always the option to adjust the level of detail of the presentation manually.

RELATED WORK
Even if not focused on the particular problem depicted in the introduction, research has been conducted, in order to enhance presentation of itineraries.

A generalization technique that is geared to hand-drawn route descriptions and tries to solve the visibility problem of minor parts of an itinerary on a constant scale factor, is presented by Agrawala and Stolte [1]. They assume that humans describe routes in a different way than systems. People always relate to their own knowledge of the environment in a route description.

In addition, humans are mainly interested in information about the main waypoints and not the connections between them. They rather neglect the length of individual roads and instead raise their visibility or specific route characteristics (e.g. a big building or a roundabout) that they consider to be relevant to the navigation process [9].

In [6], Klippel et. al. propose a formal characterization of route knowledge, that allows for communicating information on how to reach a destination (even if a specific route is not known). Therefore, changes of granularity in route directions resulting from combining elementary route information into higher-order elements (so called spatial chunking) are discussed.

The authors of that paper also point out, that if environmental features are taken into account for structuring route knowledge, a coarser perspective on the required way-finding action than simple turn-by-turn directions can be provided. Variable granularity in route directions is also focused in [10]. However, while these approaches attempt to improve the route guidance by structuring route knowledge, they disregard the individual needs of the user.

1

Most users have at least some knowledge of the vicinity they live. Although most people are familiar with their hometown or parts of it, they receive detailed route instructions from their device. A personalized granularity in route directions regarding the special knowledge of a user about the routing environment could lead to a more intelligent navigation system.

Such a comprehension of the user's knowledge about several parts of the route has been largely neglected by device manufacturers and suppliers of relevant web services, so far. The fact that such navigators would need an extended learning phase to provide customized assistance, is mostly seen as a major drawback.

To address this problem, Richter and Tomko present an approach to generate adaptive route directions generated through a dialog-based knowledge recognition process [9]. Therefore, the way-finder by default is presented with with destination descriptions, assuming that the environment is known, and can request more detailed directions using a provided dialog facility, if the currently presented information is not adequate.

We argue that a system could automatically provide user specific route directions based on a learning process that primarily is supported by a dialog-driven approach. Therefore, we act on the dialogue suggestion by Richter and Tomko, which in a first step can enhance the learning process to solve the cold start problem of completely unknown user preferences and also avoids the user from unnecessary interactions while driving.

ITINERARIES AS SEMANTIC MODELS

In order to personalize the route descriptions, we need a detailed and machine-readable model of the route in order to adapt it to the user's knowledge and preferences. Thus, we have to encode all information that may be helpful to decide whether a particular part of the route should be displayed in detail, only briefly, or not at all.

Itineraries usually are described by a set of waypoints, which represent positions between origin and target location. The idea is now to semantically enhance the waypoints in order to use the semantic information for filtering.

We therefore propose a layer model where each layer represents a degree of granularity in the route presentation. The lowest layer contains the default route directions including all details of the itinerary, as known from conventional systems. All upcoming layers show, depending on the level of abstraction, only certain parts of the route and provide the related routing instructions.

To achieve this goal, we transform the route description into a semantic model, which allows us to characterize each waypoint on the basis of its properties comprehensively. In addition to the general information of an itinerary, such as location coordinates, street name and driving instructions, a semantic description includes further information, such as

a classification of each route point on the nature and type of geographical conditions. This means that a place can either be characterized as town, city, region or even a country and the connection between two places as a street, road or motorway. This hierarchical distinction enables later filtering to distinguish the different levels of abstraction. For the transformation every route object provided by online web services like Google Maps[1] can be used.

If a user has sufficient knowledge about the environment in a particular area, only a few instructions, limited to the issue of the next motorway link and the direction of the nearest town, may be appropriate, while in areas less or not at all familiar, a detailed route guidance without any abstraction will be a better choice.

For enriching semantic itinerary models with further information, geo-services such as LinkedGeoData.org[2] can be used. Those services provide comprehensive background knowledge related to spatial features of the ways, structures and landscapes around the waypoints of an itinerary [2].

Other services that provide additional information for route enhancement are, for instance, OpenStreetMap[3], GeoNames[4], or Topocoding[5], which enables us to add the related altitude value to each waypoint. Figure 1 shows such a semantic route representation enhanced with additional information.
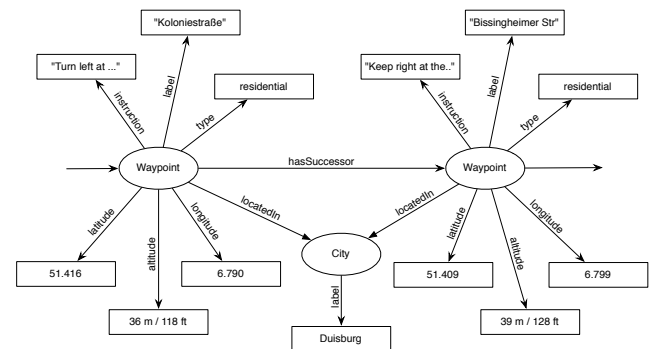

Figure 1. Semantic route representation enriched with additional information.

An itinerary that has been semantically enriched in that way, finally, facilitates the applications of particular "views" on the route. This mechanism can be used to show or hide certain waypoints and create an optimal presentation based on the users' preferences and experiences.

The navigation system could, for instance, only display prominent waypoints such as freeways (if the user already has basic knowledge of the area). In this case, the directive could simply be "Head for Freeway 1", whereas other users would receive a set of detailed instructions leading the driver to the particular freeway.

---

[1] http://maps.google.com/
[2] http://linkedgeodata.org/
[3] http://wiki.openstreetmap.org/
[4] http://www.geonames.org/
[5] http://www.topocoding.com/

The use of semantic models has different advantages compared to traditional ways routes are displayed in navigation systems:

- *S*tandardization: As information comes from various sources, each with their own formats and specifications, we need a standard to cover all these information. Semantic models are flexible enough to import all information provided by the original sources and make them accessible in a unified way (e.g. via SPARQL).

- *E*xtensibility: The characteristics of semantic models mentioned in the last paragraph allow integration of new information sources as well, regardless of their format.

- *E*ase of data processing: If the models are encoded in a standardized language like RDF or OWL, they can be queried using the *SPARQL Protocol and RDF Query Language (SPARQL)*.

- *A*dditional services: The use of standardized semantic models lays ground for future services apart from classical navigation. Recommender systems that incorporate semantic data [4], could for instance find filling stations with attractive bargains or popular restaurants on the way. Ideas for realizing such value-added services have been introduced in a german publication written by some of the authors [8].

LAYERS OF DETAIL

As an intermediate step towards a persona____ tion, we create a layered model based on the ____ so that the distinct layers reflect a particular ____ The bottom layer contains all waypoints, wh___ of-detail decreases on each layer (see Figure___ queries can be used as a filtering technique in___ or hide certain waypoints for each layer.
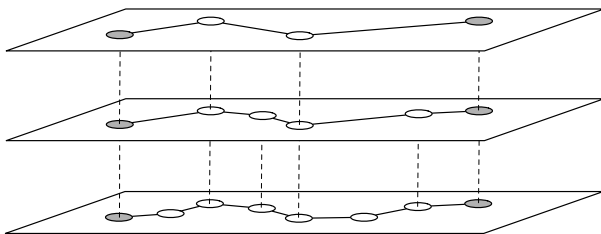

Figure 2. Particular "views" on the route of a semantically enriched itinerary.

Each layer can be seen as a "view" on the itinerary showing or hiding certain details. The base-layer corresponds to the way traditional navigators would display a route; it simply contains every single waypoint. If a higher level of abstraction is selected (either automatically or by hand), the navigator hides certain waypoints and only displays more prominent ones. If the adaptation process is supposed to be automatically instead, the level of detail can be adjusted rule-based or by other means.

ADAPTIVE ROUTE GENERATION

The layered model now allows us to switch between the levels-of-detail, such as zooming in or zooming out details of the route presentation. We provide means of manually and automatically switching between the degree of detail as well as choosing the granularity based on user profiles.

Manual Adjustment

A simple way of adjusting the presentation could be by interacting with the driver. Initially the user should be able to convey known regions dialogue based at the beginning of the guiding process, where the route has been calculated. Therefore, he can check the known parts of the itinerary step by step. Such a procedure is necessary on each guidance where no part has been marked as well known, yet. This approach is similar to the dialog-driven process described by Richter et. al. [9].

The significant deviation in our approach is that we use the dialogue initially to customize the whole route guidance on the users individual needs, while Richter provides abstract instructions by default and requires user interactions at any time the user needs more detailed ones. Nevertheless, that kind of interaction facility we will provide additionally. The user can use a simple widget such as a slider or a turning knob, which he can set up or adjust the level of detail manually.

This functionality is available at each stage of the guidance process to allow the user to react appropriately in any situ___

Figure 3. Interaction widget for manually adjust the level of detail.

User Profiles

For more sophisticated adaptation effects, dedicated user profiles cam be maintained to keep track of the user's knowledge and preferences. The system keeps track of all places and routes the user has marked as well known. It then can

provide recommendation for the levels of detail on new calculated itineraries. In this way, a user knowledge model evolves from the users interaction in a step by step manner. Figure 4 shows a schematic example of a map representing the users area knowledge, where the dark regions are assumed as well known and the lighter ones as unfamiliar.
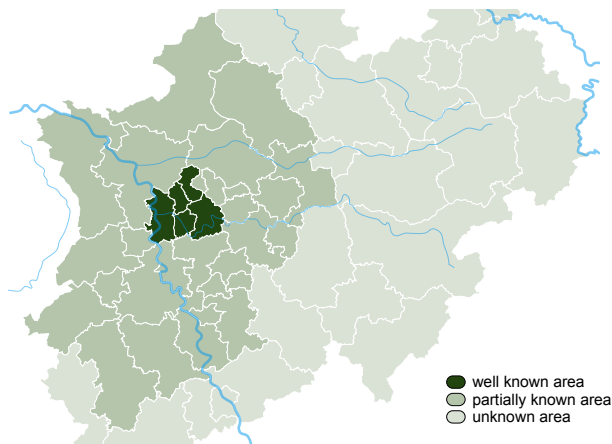

Figure 4. Schematic representation of a users individual area knowledge. Dark areas represent well known areas while bright regions are less known.

Automatic Adjustment

In order to automatically switch between the levels of detail, knowledge about the user is necessary. On the one hand, the system could incorporate information explicitly entered by the user or, alternatively, keep track of his itineraries, in order to "learn" such a profile. The first case requires user interaction, for instance by tagging certain areas on a map as "well-known" or selecting them from a list of areas.

If the system should learn and update the profile automatically based on the driver's routes, it has to keep track of the waypoints on these routes and autonomously mark them as "rather known" or "well-known". The level of detail then is based on the supposed degree of familiarity with the particular route section.

Combining explicit profile information with learning, of course, is an opportunity as well.

DISCUSSION

In this paper we presented an approach for enriching the waypoints of itineraries with semantic information, enabling a route guiding system to provide an adaptive user interface. If a driver already knows parts of the itinerary very well, the system presents the route by adapting it to the drivers preferences and experience.

During the workshop, we would like to discuss, among other issues, the following questions: What would be better? Tagging a route object with semantic information vs. converting the whole route as a semantic model? What could be alternatives for interactive definition of already known waypoints (e.g. marking them at the route planning process). How could a learning system look like that recognizes frequently used route parts or repeatedly visited places?

REFERENCES

1. M. Agrawala and C. Stolte. Rendering effective route maps: improving usability through generalization. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 241–249, New York, NY, USA, 2001. ACM.

2. S. Auer, J. Lehmann, and S. Hellmann. LinkedGeoData: Adding a spatial dimension to the Web of Data. *The Semantic Web-ISWC 2009*, pages 731–746, 2009.

3. J. Hu, I. Kaparias, and M. Bell. Spatial econometrics models for congestion prediction with in-vehicle route guidance. *Intelligent Transport Systems, IET*, 3(2):159 –167, 2009.

4. T. Hussein, T. Linder, W. Gaulke, and J. Ziegler. Context-aware recommendations on rails. In *Workshop on Context-Aware Recommender Systems (CARS-2009) in conjunction with the 3rd ACM Conference on Recommender Systems (ACM RecSys 2009)*, New York, NY, USA, 2009.

5. Institut für Demoskopie Allensbach. ACTA 2010 - Innovationen treiben die Märkte. Presentation, October 2010.

6. A. Klippel, S. Hansen, K. Richter, and S. Winter. Urban granularities—a data structure for cognitively ergonomic route directions. *GeoInformatica*, 13(2):223–247, 2009.

7. T. Kujala. Efficiency of visual time-sharing behavior: the effects of menu structure on poi search tasks while driving. In *AutomotiveUI '09: Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 63–70, New York, NY, USA, 2009. ACM.

8. D. Münter, T. Hussein, and W. Gaulke. Kontextabhängige empfehlung von services zur intelligente navigationsunterstützung. In *Proceedings of the 1st German Workshop on Human Service Interaction*, 2010.

9. K. Richter, M. Tomko, and S. Winter. A dialog-driven process of generating route directions. *Computers, Environment and Urban Systems*, 32(3):233–245, 2008.

10. T. Tenbrink and S. Winter. Variable granularity in route directions. *Spatial Cognition & Computation*, 9(1):64–93, 2009.

11. J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 220–232, New York, NY, USA, 2007. ACM.

SemSor: Combining Social and Semantic Web to Support the Analysis of Emergency Situations

Philipp Heim
Institute for Visualization and
Interactive Systems (VIS)
Universitätsstraße 38
Stuttgart, Germany
philipp.heim@vis.uni-
stuttgart.de

Dennis Thom
Institute for Visualization and
Interactive Systems (VIS)
Universitätsstraße 38
Stuttgart, Germany
dennis.thom@vis.uni-
stuttgart.de

Thomas Ertl
Institute for Visualization and
Interactive Systems (VIS)
Universitätsstraße 38
Stuttgart, Germany
thomas.ertl@vis.uni-
stuttgart.de

ABSTRACT

In this paper we introduce SemSor, a system developed especially for the analysis of emergency situations. It constantly collects information from sources of the Social Web, maps it to unique resources in the Semantic Web and uses the annotated information as basis for the situation analysis. If an emergency situation needs to get analyzed, four steps are required: First, all information that is already known about this situation must be entered in the SemSor-GUI. Second, the entered information needs to be mapped to resources in the Semantic Web. Third, using these resources as starting nodes, a spreading activation is applied along the relationships within the Semantic Web to find relevant Social Web information. And fourth, the newly identified information is visualized according to different dimensions and can be filtered and explored by the user. In an iterative process, new insights can be used to refine the query and thus improve the activated information until a comprehensive analysis of even complex situations is possible.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Indexing methods; H.3.3 [**Information Search and Retrieval**]: Information filtering, query formulation, relevance feedback; H.5.2 [**User Interfaces**]: Graphical user interfaces (GUI)

Keywords

Social Web, Web 2.0, Semantic Web, Social Semantic Web, situation analysis, spreading activation, interactive information retrieval

1. INTRODUCTION

Analyzing emergency situations is difficult in cases where either the agent who does the analysis is not close by and thus is not able to directly hear or see what is going on, or the situation itself is distributed in space or time so that one single person alone has difficulties in getting an overview. These cases hold true, e.g., in *emergency operation centers (EOCs)*, where neither the agent who receives an emergency call is onside nor the person who has made the call has usually a comprehensive picture of the situation. However, having a comprehensive picture is especially important for the analysis of emergency situations in order to take the right actions and thus prevent all kinds of damage.

To overcome the difficulties in analyzing emergency situations, we propose an approach that combines the advantages of the Social Web with those of the Semantic Web. The idea is to scan Social Web entries, semantically annotate their content and use spreading activation to find exactly those entries that are useful for the analysis of a specific emergency situation. The idea of combining Social and Semantic Web to a *Social Semantic Web* has already been described e.g. in [10] and implemented in many applications, e.g. within the *WeKnowIt*-Project [6]. Also tools have been developed that use this idea to support the analysis of emergency situations [8, 13]. In these tools the found Social Web entries are often arranged on a map to provide an overview of the geographical extent, e.g. on the *Interactive Fire Map* [3], or to extract relevant information via geographical filters [14]. Also popular are timelines that order Social Web entries according to the date of their creation and thus support an understanding of the chronology of events [15]. However, none of these approaches use spreading activation to find semantically related Social Web entries automatically.

The general idea of spreading activation in semantic graphs has first been introduced in [9]. Initially a set of starting nodes is labeled with activation energy, which then is iteratively propagated to other nodes that are linked to the starting nodes. Links can be weighted in order to control the spreading of energy. This can be used, for instance, in recommender systems to adapt the content of a web site to the current context of its visitors. Next to the users' concrete information needs, contextual information like location, time, role, or weather conditions can be used to spread the activation differently and thus to find information that is relevant with respect to a specific context [12]. In addition, every user action can lead to refined link weights and activation energies to account for individual preferences and interests. However, this requires the semantic graph to be stored locally or on a server with write permissions in order to be able to adapt the weights and activation energies accordingly.
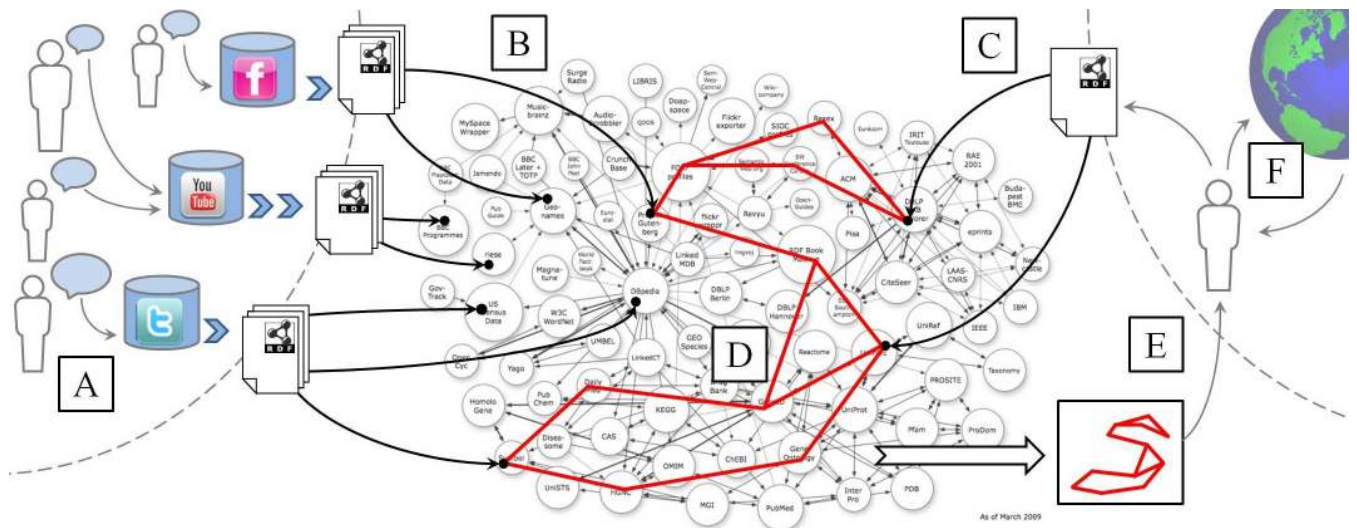
Figure 1: SemSor architecture: Social Web entries are constantly crawled (A) and mapped to semantic resources (B). If a situation has to be analyzed, all known information also needs to get mapped to semantic resources (C), which function as starting nodes for the spreading activation (D). The found Social Web entries are visualized and can be explored and filtered by the user (E). Gained insights, from the visualization or from external sources (F), can then iteratively be used to improve the activated information.

In this paper we introduce an approach that applies spreading activation in external semantic datasets and thus saves storage space and calculating capacities. Datasets in the *LOD cloud* [2] are accessed via *SPARQL* [5] queries to trigger the spreading activation and thus to find semantically related resources. Besides the low system requirements, the two main advantages of an outsourced spreading activation approach are: 1) The datasets are always up-to-date; no complicated methods for updating local copies are required. 2) Semantic relationships of all kinds and domains are used to activate relevant information; spreading activation is not restricted to a predefined set of resources, e.g. resources of a certain domain, but can include all domains contained in the LOD cloud. With the SemSor system, we present an prototypical implementation of our approach that facilitates the extraction of community information relevant to analyze a certain emergency situation. Even though the spreading activation takes place externally, the user can rate the relevance of the found Social Web entries to refine the search query and thus change the activation values until a thorough analysis can be achieved.

In the following we first describe the general SemSor architecture with all the components and steps that are required for the analysis of emergency situations and provide further details to each of the steps afterwards. This includes the crawling and annotating of Social Web entries, the initial query formulation, the spreading activation, the visualization and filtering, and the interactive query refinement. At the end of the paper a conclusion and an outlook on future work is given.

2. SYSTEM ARCHITECTURE

The SemSor System constantly scans Social Web sources, e.g. *Twitter*, *Flickr* and *YouTube*, for new entries (Fig. 1, A) and semantically annotates their textual content. Terms with a distinct meaning, e.g. geographical or temporal ref-

erences, are therefore mapped to unique semantic resources of datasets in the LOD cloud, e.g. *DBpedia* [7] or *GeoNames* [1] (Fig. 1, B). Thus the system automatically creates a machine readable representation of the semantic that is contained in the found Social Web entries, which can later on be used to support the analysis.

Once a certain situation needs to be analyzed (Fig. 1, F), e.g. because of an incoming emergency call, everything that is known about this situation must also be mapped to unique semantic resources (Fig. 1, C). These resources are then used as starting nodes for the spreading activation [9] that is applied to find all semantic resources that might be of relevance (Fig. 1, D). In a next step, all the Social Web entries that have been annotated with at least one of the activated resources are collected and form the result set, which is presented to the user via multiple views (Fig. 1, E). The result set can interactively be explored and filtered by the user in order to gain new insights about the situation. Gained insights or news from external sources, e.g. from the first responders, (Fig. 1, F) can then iteratively be used to refine the search query, thereby activate new resources in the Semantic Web and thus improve the situation analysis. Due to the possibility to iteratively refine the search query, humans and computers can co-operate in this task.

3. CRAWLING AND ANNOTATING SOCIAL WEB ENTRIES

In order to use information that is contained in Social Web entries to support the analysis of emergency situations, the entries first have to be extracted and annotated by the crawler component of SemSor. Even though the Social Web contains a huge amount of data, only a minimum of this information needs to get stored in the SemSor database. In a first step, a broad multitude of Social Web data gets collected and evaluated according to a preconfigured metric that determines the *a priori relevance* of each individual

entry. Within this metric, different properties of an entry like its source, the date and time of its creation as well as location-based data get extracted and serve as basis to calculate a weighted importance rating. The weights of the metric can be configured according to the individual needs of its users (e.g. a specific emergency response team) and provide a basic means to decide, which entries should be kept and which can be deleted if computational- or storage-resources become short. Following the collection and a priori evaluation, the Social Web entries are analyzed and certain terms in their textual contents are automatically assigned to unique resources in datasets in the LOD cloud by using services like e.g. *OpenCalais* [4]. Once new entries have been registered and evaluated, only their URLs and the URIs of the assigned semantic resources have to be kept for subsequent steps.

4. INTERACTIVE SITUATION ANALYSIS

The SemSor system supports the whole situation analysis process, including the initial query formulation, the search for relevant information via spreading activation, the visualization and filtering of the results as well as mechanisms to iteratively refine the query.

4.1 Initial Query Formulation

The method of query is based on common question schemes of emergency calls according to relevant aspects of a situation: "What has happened?", "Where did it happen?" and "Who is involved?" (Fig. 2, B). The agent is supposed to provide approximate answers to at least some of these questions and can further substantiate his query by providing boundaries for the temporal and spatial extent of the situation. In this process, the agent is assisted by an adaptive autocomplete feature, which will try to interactively map given search terms to resources in the Semantic Web. Throughout this process the definitions of proposed resources are provided in pop-up windows, e.g. corresponding *Wikipedia* articles (Fig. 2, A), to help users especially in the disambiguation of ambiguous input terms. Based on this procedure the system is able to get a reliable handle onto the relevant nodes in the Semantic Web. During an emergency situation, like e.g. the 2010 Haiti earthquake, disaster agents can obtain a general overview of the situation by performing a broad search on keywords like "Earthquake" and "Haiti". Based on the interactive mapping of search terms to semantic resources the query is annotated by SemSor and connected with resources in the Semantic Web. The nodes of this framework serve as the initially activated nodes in the spreading activation procedure.

4.2 Spreading Activation

Based on the initial activation of the user defined staring nodes (Fig. 3, A), a homogeneous spreading activation is applied along the relationships between the resources in the datasets. In order to automatically activate semantically related resources that might also be relevant for the situation analysis, the activation happens along the instance-relationship layer (Fig. 3, B) as well as the class-relationship layer (Fig. 3, C). Thus the resulting set of Social Web entries is not limited to those containing references to one of the user defined starting nodes only, but also includes entries referring to resources that are within a distinct semantic radius around the user defined starting nodes (Fig. 3, D);
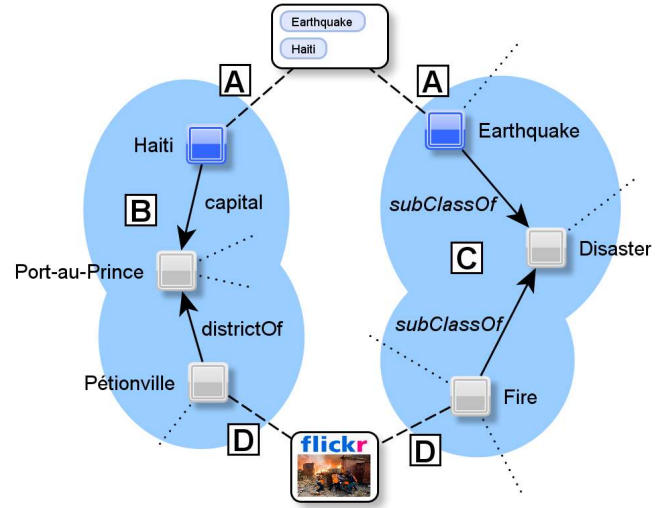

Figure 3: Search terms are interactively mapped to resources in semantic datasets (A) that function as starting nodes for the spreading activation that is applied along the links within these datasets (B and C). As a result, semantically related resources get activated and thus Social Web entries annotated with at least one of them get found (D).

this facilitates finding information that is relevant for the analysis of a certain emergency situation.

The spreading activation in SemSor is implemented mostly as a remote process. On the client side, the process is only triggered and controlled but is run completely within external datasets on server side. Therefore SPARQL queries are sent to the datasets to find resources related to the starting nodes, which are then scored according to the semantical and topological properties of their relationships.

Related resources are found based on an approach described in [11]. Taking the starting nodes as roots, a *breadth-first search (BFS)* is applied to find all resources that are related to one of the starting nodes up to a predefined depth threshold. The depth threshold defines the number of nodes that are allowed between a starting node and a resource that can be activated. Thus having e.g. a depth threshold equal null restricts the activation radius to resources that are directly connected to one of the starting nodes. For each related resource that is found, the algorithm checks whether Social Web entries have been crawled that are assigned to it (e.g. the Flickr entry is assigned to Fire in Fig. 3).

All those Social Web entries are then activated according to topological and semantical aspects and thereby scored. In our implementation the extent of activation depends on three aspects: 1) The length of the relationship, e.g. the length between the Flickr entry and the starting nodes is two, 2) the connection types within the relationships, e.g. "district of" or "capital", and 3) the classes of the intermediate resources, e.g. "Port-au-Prince" is a city. It is also possible to define certain connection types or classes that should not be used to spread the activation, which is useful if someone is not interested in relationships that contain certain instances or connections.

Figure 2: SemSor GUI: Search terms are interactively mapped to semantic resources (A) that together form the search query (B). Relevant information from Social Web sources is found automatically and shown in a list (C). Single entries can be examined in detail (D) and filters can be formulated according to various dimensions, e.g. spatial filters (E) or temporal filters (G).

4.3 Visualization and Filtering

Based on their activation, the Social Web entries are visualized in the SemSor GUI. While the search still continues, all Social Web entries that were already discovered by the activation are presented through the result browser in different user-selectable views (e.g. tabular, map and statistic view). Every view provides the opportunity to obtain pictures, videos and other user generated content related to the situation (Fig. 2, D). The standard view is a listing of entries sorted by their individual semantic relevance (Fig. 2, C). In order to get an overview of the spatial distribution of possibly relevant entries, the agent can view them on a map that can also be used to formulate geographical constraints, e.g. to show only results that refer to a certain geographic region (Fig. 2, E). To further explore the set of results a time line offers an overview over the temporal distribution of events (Fig. 2, G) and allows to formulate temporal constraints, e.g. to show only results that refer to a specific period of time. To further explore the quality and diversity of results the agent has the opportunity to analyze diagrams that show the composition of chosen result-subsets by author, location or tag-categories. By acquiring these initial impressions, the agent can further asses the nature and extent of the situation and initiate subsequent steps.

4.4 Interactive Query Refinement

At this point the advantages of the interactive features in SemSor come into play. If the resulting set generated by the initial spreading activation (Fig. 4, A and B) and diminished by the user defined filters is yet not sufficient, the agent can further refine and expand his initial query by rating single result items on a continuous scale (Fig. 4, C)

and by dragging additional tags from the items to the query fields. High ratings of some Social Web entries can then lead to the activation of semantic resources which are directly connected to those entries (Fig. 4, D). Thus after rating the items, the agent is given the possibility to restart his search, but this time, the spreading activation will execute starting also from the newly activated items (Fig. 4, E). Through this "pollination" and subsequent activation of remote nodes in the semantic graph the agent is given the chance to discover relevant regions and new Social Web entries that were not included or even near his initial query (Fig. 4, F). Based on the filters and the interaction procedure, the agent is able to cope with the enormous flood of Social Web data that can be found in connection with emergency situations and make beneficial use of them.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we described an approach that combines the Social and the Semantic Web in order to support the analysis of emergency situations. Exploiting information from the Social Web is especially useful when: (1) the situation is distributed in space or time, (2) first responders are not on site, or (3) the situation cannot or only partly be observed; e.g. this can be the case if a situation takes place within a building or is hidden behind some obstacles.

The automatic semantic annotation of entries in the Social Web as well as the interactive mapping of entered search terms to unique resources in the Semantic Web allows information to be found not only by string matching but also according to its meaning. Therefore spreading activation is applied in external semantic datasets that offer both up-to-date and comprehensive information on all kinds of topics.
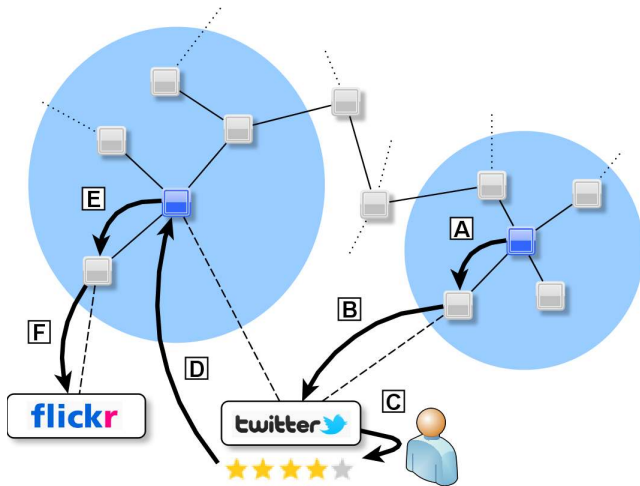
Figure 4: Once an entry has been found via spreading activation (A and B), the agent can rate it as relevant (C) and thus indirectly expand the search query (D). This leads to the activation of other nodes (E) and thus can produce Social Web entries as result set that are only distantly related to the user defined search query (F).

Because of the outsourced spreading activation and the minimum information that is needed to store the annotated Social Web entries – an entry is represented by its URL and the URIs of the annotated resources only – SemSor is able to handle the huge amounts of available data and find information relevant for the analysis of a certain emergency situation. The found information is visualized in multiple views and can be explored and filtered by the agent based on individual information needs. If the result set is not yet sufficient, the query can interactively be expanded or narrowed down. Single entries can be rated as relevant or as irrelevant which changes the starting nodes and thus can result in other entries to be found by the spreading activation. The query can be refined until a sufficient analysis of the emergency situation is possible.

In its current implementation, the SemSor system is most suitable to analyze current or past emergency situations (cp. Fig. 2). Since many people use the Social Web to comment on emergency situations, relevant information is available even while a situation is happening. Together with the fact that only seconds are required from the time a new comment is uploaded to when it can be found in SemSor, it is already possible to facilitate the analysis of current and past emergency situations.

However, analyzing current or past emergency situations often cannot prevent them from happening. In order to prevent emergency situations a preventive analysis is required. First signs of a forthcoming emergency situation need to be detected and interpreted in the right way so that the right actions can be initiated. Besides the agent triggered search, this would require SemSor to automatically scan the Social Web entries for new topics and trends and iteratively produce an overview of the current situation. If certain topics get popular or unusual changes can be detected an alarm could be raced automatically that could force an agent to check the situation and decide on the right actions to pre-

vent a possible emergency situation from happening.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] Geonames. `http://www.geonames.org/`, 2006.
[2] Linking open data community project (lod). `http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`, 2007.
[3] San diego wildfires 2007 interactive fire map. `http://www.signonsandiego.com/firemap/`, 2007.
[4] Opencalais. `http://www.opencalais.com/`, 2008.
[5] Sparql protocol and rdf query language. `http://www.w3.org/TR/rdf-sparql-query/`, 2008.
[6] Weknowit project. `http://www.weknowit.eu`, 2010.
[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: a nucleus for a web of open data. In *Proc. of the 6th Int. Semantic Web Conf. (ISWC'08)*, volume 4825 of *LNCS*, pages 722–735. Springer, 2008.
[8] J. Borsje, L. Levering, and F. Frasincar. Hermes: a semantic web-based news decision support system. In *Proc. of the 2008 ACM symposium on Applied computing*, SAC '08, pages 2415–2420, New York, NY, USA, 2008. ACM.
[9] A. M. Collins and E. F. Loftus. A spreading-activation theory of semantic processing. *Psychological Review*, 82(6):407 – 428, 1975.
[10] T. Gruber. Collective knowledge systems: Where the social web meets the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6:4–13, 2008.
[11] P. Heim, S. Hellmann, J. Lehmann, S. Lohmann, and T. Stegemann. Relfinder: Revealing relationships in rdf knowledge bases. In *Proc. of the 4th Int. Conf. on Semantic and Digital Media Technologies (SAMT 2009)*, pages 182–187. Springer, 2009.
[12] T. Hussein and J. Ziegler. Adapting web sites by spreading activation in ontologies. In *Proc. of the Int. Workshop on Recommendation and Collaboration (ReColl '08)*, 2008.
[13] N. Ireson. Local community situational awareness during an emergency. In *Proc. of the IEEE Int. Conf. on Digital Ecosystems and Technologies (DEST 2009)*, pages 49–54. IEEE, 2009.
[14] C. P. Julio and C. A. Iglesias Fernández. Disasters 2.0: Application of web 2.0 technologies in emergency situations. In *Proc. of the 6th Int. ISCRAM Conf.*, Gothenburg, Sweden, 2009.
[15] S. B. Liu and L. Palen. Spatiotemporal mashups: A survey of current tools to inform next generation crisis support. In *Proc. of the 6th Int. ISCRAM Conf.*, Gothenburg, Sweden, 2009.

Interactive News Video Recommendation: An Example System

Frank Hopfgartner
International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA, 94704
fh@icsi.berkeley.edu

ABSTRACT

This position paper introduces a recommender system which has been developed to study research questions in the field of news video recommendation and personalization. The system is based on semantically enriched video data and can be seen as an example system that allows research on semantic models for adaptive interactive systems.

1. INTRODUCTION

In recent years, the amount of multimedia content available to users has increased exponentially. This phenomenon has come along with (and to much an extent is the consequence of) a rapid development of tools, devices, and social services which facilitate the creation, storage and sharing of personal multimedia content. A new landscape for business and innovation opportunities in multimedia content and technologies has naturally emerged from this evolution, at the same time that new problems and challenges arise. In particular, the hype around social services dealing with visual content, such as YouTube or Dailymotion has led to a rather scattered publishing of video data by users worldwide [8]. Due to the sheer amount of large data collections, there is a growing need to develop new methods that support the users in searching and finding videos they are interested in.

Video retrieval is a specialization of information retrieval (IR), a research domain that focuses on the effective storage and access of data. In a classical information retrieval scenario, a user aims to satisfy their information need by formulating a search query. This action triggers a retrieval process which results in a list of ranked documents, usually presented in decreasing order of relevance. The activity of performing a search is called the information seeking process. A document can be any type of data accessible by a retrieval system. In the text retrieval domain, documents can be textual documents such as emails or websites. Image documents can be photos, graphics or other types of visual illustrations. Video documents consist of a set of audio-visual signals and accompanying metadata. The audio-visual features can be described by low-level feature descriptors, the main description standard being MPEG-7.

Retrieving videos using low-level features is, due to the Semantic Gap [18], a challenging approach. An analysis of state-of-the-art research on video retrieval indicates that content-based video retrieval performance is still far away from their textual counterparts [7]. An interesting approach to narrow this performance gap is to further enrich video documents using external data sources, called metadata. Blanken et al. [4] list three types of metadata: (1) Descriptive Data, (2) Text Annotations and (3) Semantic Annotation. All approaches aim to provide annotations in textual form that allow to bridge the Semantic Gap. Fernández et al. [9], for instance, have shown that ontology-based search models that exploit semantic annotations can outperform classical information retrieval models at a web scale. The advantage of these models is that external knowledge is used to set the content into their semantic context.

In [10], we introduced a news video recommender system which relies on such semantic annotations. The system captures daily broadcasting news, and segments the bulletins into semantically related news stories. DBpedia is exploited to set these stories into context. DBpedia is a structured representation of Wikipedia [2]. This semantic augmentation of news stories is used as the backbone of our news video recommendation. Our first hypothesis was that implicit relevance feedback can be used to create appropriate long-term user profiles. Implicit relevance feedback refers to user interactions that are performed implicitly during a search session, such as clicking a search result or spending time to read/view a document. We introduced an implicit user modeling approach which automatically captured users' evolving information needs, representing interests in a dynamic user profile. Another research question was to study whether the selection of concepts in a generic ontology can be used for accurate news video recommendations. Therefore, we introduced our approach of exploiting DBpedia to set concepts of news stories into their semantic context. As our evaluation indicates, semantic recommendations can successfully be employed to improve the recommendation quality.

While we evaluated within this work the underlying personalization technique, which takes advantage of an ontology, the impact of the adaptive presentation of the recommendations and search results, i.e. the interface design, has not

been evaluated yet. Given a well-evaluated backend which relies on Semantic Web technologies, we argue in this position paper that the introduced personalization system can be seen as an exemplar system which allows for studying the research questions that are within the scope of this workshop. After introducing the research domain in Section 2, we illustrate in Section 3 how users can use the system to receive frequent news video recommendations that match their personal interests. In Section 4, we introduce the interface of prior mentioned system, which is required to visualize semantically enriched video data. Section 5 discusses how this system can be used as an example to study semantic models for adaptive interactive systems.

2. SEMANTIC NEWS VIDEO RECOMMENDATION

When interacting with a video retrieval system, users express their information need in search queries. The underlying retrieval engine then retrieves relevant results to the given queries. A necessary requisite for this IR scenario is to correctly interpret the users' information need. As Spink et al. [19] indicate though, users very often are not sure about their information need. One problem they face is that they are often unfamiliar with the data collection, thus they do not exactly know what information they can expect from the corpus [17]. Further, Jansen et al. [12] have shown that video search queries are rather short, usually consisting of approximately three terms. Considering these observations, it is hence challenging to satisfy users' information needs, especially when dealing with ambiguous queries. Triggering the short search query "Victoria", for example, a user might be interested in videos about cities called Victoria (e.g. in Canada, United States or Malta), landmarks (e.g. Victoria Park in Glasgow or London), famous persons (e.g. Queen Victoria or Victoria Beckham) or other entities called Victoria. Without further knowledge, it is a demanding task to understand the users' intentions. Interactive information retrieval aims at improving the classic information retrieval model by studying how to further engage users in the retrieval process, in a way that the system can have a more complete understanding of their information need. Thus, aiming to minimize the users' efforts to fulfill their information seeking task, there is a need to personalize search. In a web search scenario, Mobasher et al. [14] define personalization as "any action that tailors the Web experience to a particular user, or a set of users". Another popular name is adaptive information retrieval, which was coined by Belew [3] to describe the approach of adapting, over time, retrieval results based on users' interests.

Most of the approaches that follow the interactive information retrieval model are based on relevance feedback techniques [17]. Relevance feedback (RF) is one of the most important techniques within the IR community. An overview of the large amount of research focusing on exploiting relevance feedback is given by Ruthven and Lalmas [16]. The principle of relevance feedback is to identify the user's information need and then, exploiting this knowledge, adapting search results. Rocchio [15] defines relevance feedback as follows: The retrieval system displays search results, users provide feedback by specifying keywords or judging the relevance of retrieved documents and the system updates the results by incorporating this feedback. The main benefit

of this approach is that it simplifies the information seeking process, e.g. by releasing the user from manually reformulating the search query, which might be problematic especially when the user is not exactly sure what they are looking for or does not know how to formulate their information need. Two types of relevance feedback exist: explicit and implicit feedback. While explicit RF models rely on users permanently providing relevance information about documents they retrieved, implicit RF models rely on automatically mining user interaction data. The main advantage is that this approach delivers the user from providing explicit feedback.

Most personalization services rely on users explicitly specifying preferences. However, users tend not to provide constant explicit feedback on what they are interested in. In a long-term user profiling scenario, this lack of feedback is critical, since feedback is essential for the creation of such profiles. Considering that each interface feature is designed to allow users to either retrieve or explore document collections, we hypothesized in [10] that the users' interactions with these features can be exploited as implicit relevance feedback. We introduced a news video recommender system which automatically generates personalized multimedia news that cover topics of the users' long-term interests.

Defining the technical conditions for such recommender systems, we argued that the creation of a private news video collection is required, consisting of up-to-date news bulletins from different broadcasting stations. Further, we argued that semantic web technology can be exploited to link concepts in the news broadcasts and suggested a categorization of stories into broad news categories. From a user profiling point of view, these links and categories can be of high value to recommend semantically related transcripts, hence creating a semantic-based user profile. For example, a user could show interest in a story about the sunset at the Greek island Santorini. The story transcript might contain the following sentence:

> "This is Peter Miller, reporting live from Santorini, Greece, where we are just about to witness one of the most magnificent sunsets of the decade. [...]".

If the same user enjoys travel with emphasis on warm Mediterranean sites, he/she might also be interested in a report about the Spanish island Majorca. For example, imagine the following story:

> "Just as every year, thousands of tourists enjoy their annual sun bath here in Majorca. [...]".

An interesting research question is how to identify whether this story matches the user's interests. Lioma and Ounis [13] argue that the semantic meaning of a text is mostly expressed by nouns and foreign names, since they carry the highest content load. Indeed, most adaptation approaches rely on these terms to personalize retrieval results, e.g. by performing a simple query expansion. The two example stories, however, do not share similar terms. A personalization

technique exploiting the terms only would hence not be able to recommend the second story. However, linking the concepts of the transcripts using DBpedia reveals the semantic context of both stories. It becomes evident that both stories are about two islands in the Mediterranean Sea. Exploiting this link could hence satisfy the user's interest in warm Mediterranean Sites. We therefore proposed to set news broadcasts into their semantic context by exploiting the large pool of linked concepts provided by DBpedia.

Having established a semantically annotated data collection, the recommender system can be operated on a regular basis to retrieve news stories that match the user's interests. In the next section, we illustrate a typical use-case that illustrates the use of the exemplar system.

3. USE-CASE SCENARIO

In the previous section, we provided a brief summary of the research challenges that have been tackled in [10]. Users can interact with this system on a regular basis, e.g. over several weeks, to satisfy their information need, allowing for longitudinal user studies where the system can be evaluated. The following example depicts a typical use-case scenario:

> "Imagine a user who is interested in multiple news topics. They registered with a news recommender system with a unique identifier. For a period of several months, they log into the system, which provides them access to the latest news video stories of the day. On the system's graphical interface, they have a list of the latest stories which have been broadcast on two national television channels. They now interact with the presented results and logs off again. On each subsequent day, they log in again and continue the above process."

In this scenario, a user frequently uses the system to gather latest news. The interface has been designed to adapt its content based on users' personal interests by employing the semantic context of the data collection. Each time, he/she interacts with the video documents which have been displayed by the graphical user interface, he/she leaves a "semantic fingerprint" of their interests. Based on this fingerprint, more video documents are identified by exploiting the semantic link between the video documents in the collection. Hence, each time the user interacts with retrieval results, other related videos are identified and displayed. A long-term user study focusing on evaluating the performance of different recommendation techniques has been introduced in [11].

While this evaluation is focused on the recommendation techniques, a thorough evaluation of the interface has not been done yet. An overview over the interface is given in the next section.

4. INTERFACE DESIGN

Figure 1 shows a screenshot of the adaptive news video retrieval interface which was used within the study. It can be split into three main areas: Search queries can be entered in the search panel on top, results are listed on the right side and a navigation panel is placed on the left side of the interface. When logging in, the latest news will be listed in the results panel. Search results are listed based on their relevance to the query. Since we are using a news corpus, however, users can re-arrange the results in chronological order with latest news listed first. Each entry in the result list is visualized by an example key frame and a text snippet of the story's transcript. Keywords from the search query are highlighted to ease the access to the results. Moving the mouse over one of the key frames shows a tool tip providing additional information about the story. A user can get additional information about the result by clicking on either the text or the key frame. This will expand the result and present additional information including the full text transcript, broadcasting date, time and channel and a list of extracted named entities. In the example screenshot, the third search result has been expanded. The shots forming the news story are represented by animated key frames of each shot. Users can browse through these animations either by clicking on the key frame or by using the mouse wheel. This action will center the selected key frame and surround it by its neighboring key frames. The user's interactions with the interface are exploited to identify multiple topics of interests. On the left hand side of the interface, these interests are presented by different categories, i.e. those news categories that the user showed interest in during previous search sessions.

Summarizing, the interface provides access to different news categories in which the user showed interest in. These interests can adapt over time, i.e. when a user shows interest in a certain news aspect right now, this aspect might already be irrelevant in a few days. Imagine, for example, a user who has shown high interest in any news regarding the FIFA Soccer World Cup. Just a few days after the end of the tournament, the user's interest might drop to a minimum again. Our interface serves this evolving need by automatically updating the categories in which the user showed the most interest in during the last sessions. The evolving interest is modeled by applying the Ostensive Model [6], which provides a decay function that aligns a higher weighting to more recent user interests.

5. DISCUSSION AND CONCLUSION

Above description reveals that the interface has been designed to visualize news videos that match users' interests. The categorization of these interests is highly user-centric. The interface adapts its content, i.e. both categories on the left hand side and news videos on the right hand side based on the users' previous interactions. Even though the recommendation technique relies on interlinked data, the interface itself does not support filtering or browsing the data accordingly.

As mentioned before, this constraint is due to the different focus of the research, which was aiming at studying recommendation techniques rather than adaptive interface designs. Nevertheless, given the support of semantically enriched video data, we argue that the system can be seen as an example framework which enables to study such interface features. Example improvements include visualizing story interlinking by using a hyperbolic tree, as has been

Figure 1: News Video Recommender Interface

introduced by Bürger et al. [5]. In their Smart Content Factory, each document in the index has been enriched with semantic information, i.e. places mentioned in the transcript are matched with a generic geography thesaurus. Such tree would allow users to browse the video collection based on the semantic content of each video. Another improvement could be to provide thesaurus supported query auto-completion features as shown by Amin et al. [1]. This would allow users to get an idea about the collection based on the query suggestions.

Acknowledgment

6. REFERENCES

[1] A. Amin, M. Hildebrand, J. van Ossenbruggen, V. Evers, and L. Hardman. Organizing suggestions in autocompletion interfaces. In *ECIR'09: Proceedings of the 31st European Conference on IR Research, ECIR 2009, Toulouse, France*, pages 521–529, 2009.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proc. 6th Int. Semantic Web Conf.*, pages 722–735. Springer Berlin / Heidelberg, 11 2007.

[3] R. K. Belew. Adaptive information retrieval: using a connectionist representation to retrieve and learn about documents. *SIGIR Forum*, 23(SI):11–20, 1989.

[4] H. M. Blanken, A. P. de Vries, H. E. Bok, and L. Feng. *Multimedia Retrieval*. Springer Verlag, Heidelberg, Germany, 1 edition, 2007.

[5] T. Bürger, E. Gams, and G. Güntner. Smart content factory: assisting search for digital objects by generic linking concepts to multimedia content. In *Proc. HT*, pages 286–287. ACM, 2005.

[6] I. Campbell and C. J. van Rijsbergen. The ostensive model of developing information needs. In *Proc. Library Science*, pages 251–268, 1996.

[7] M. G. Christel. Establishing the utility of non-text search for news video retrieval with real world users. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 707–716, New York, NY, USA, 2007. ACM.

[8] S. J. Cunningham and D. M. Nichols. How people find videos. In *Proc. 8th ACM/IEEE-CS Joint Conference on Digital libraries*, pages 201–210, New York, NY, USA, 2008. ACM.

[9] M. Fernández, V. López, M. Sabou, V. Uren, D. Vallet, E. Motta, and P. Castells. Using TREC for cross-comparison between classic IR and ontology-based search models at a Web scale. In

SemSearch'09, 4 2009.

[10] F. Hopfgartner and J. M. Jose. Semantic user modelling for personal news video retrieval. In *MMM*, pages 336–346, 2010.

[11] F. Hopfgartner and J. M. Jose. Semantic user profiling techniques for personalised multimedia recommendation. *Multimedia Systems*, 16(4):255–274, 2010.

[12] B. J. Jansen, A. Goodrum, and A. Spink. Searching for multimedia: analysis of audio, video and image web queries. *World Wide Web*, 3(4):249–254, 2000.

[13] C. Lioma and I. Ounis. Examining the Content Load of Part of Speech Blocks for Information Retrieval. In *ACL'06: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia*, 2006.

[14] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[15] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART retrieval system: experiments in automatic document processing*, pages 313–323, Englewood Cliffs, USA, 1971. Prentice-Hall.

[16] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(2):95–145, 2003.

[17] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in information retrieval*, pages 355–364, 1997.

[18] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[19] A. Spink, H. Greisdorf, and J. Bateman. From highly relevant to not relevant: examining different regions of relevance. *Inf. Process. Manage.*, 34(5):599–621, 1998.

A Context-Aware Proactive Controller for Smart Environments

Frank Krüger*   Gernot Ruscher   Sebastian Bader   Thomas Kirste

Universität Rostock,
Albert-Einstein-Str. 21,
18059 Rostock, Germany
*corresponding author: frank.krueger2@uni-rostock.de

ABSTRACT

In this paper we describe an *implicit* user interface for smart environment control: We make our system *guess* how to assist the user(s) proactively. Our controller is based on two formal descriptions: One that describes user activities, and another that specifies the devices in the environment. Putting both together, we can synthesize a probabilistic model, the states of which resemble activities performed by the user(s) and are annotated with sequences of device actions, with the latter to be executed in cases particular activities have been recognized. The resulting system is purely reactive and can be executed in real time.

Categories and Subject Descriptors

H.5 [**User Interfaces**]: Input Devices and Strategies

General Terms

Theory

Keywords

intention recognition, HMM, planning, smart environments

1. INTRODUCTION

As computers become smaller and smaller, the vision of ubiquitous computing becomes true. At the same time, smart environments contain a large number of devices and become thus more and more complex. Thus, configuration as well as correct usage gets more time consuming and error prone. Exploring new ways to control these *invisible* devices is a challenge addressed by current research [2]. Our approach is to create an entirely reactive system to control all devices of the environment by inferring the intentions of the user. The system gives support by controlling the devices the way the user would do to achieve his *goals*. We use a semantic modeling of the user and the environment to assure that the support is sound and complete, in the sense that the environment is able to support the user correctly in every recognizable situation.

To proactively support users in instrumented environments, we need to infer their intentions, the goals behind their current activities. Here we use a rather technical notion of intention: given descriptions of complex actions, like giving a presentation or preparing a meal. If we detect the user performing some sub-tasks of these complex action, we assume that his goal is to perform the complex action completely. A controller such as described requires all calculation to be executed in realtime. It is therefore necessary to move time consuming operations like planning processes from runtime to compile time. Thus, we can create a purely reactive controller with time-bounded complexity, able to control the environment in every possible situation.

As illustrating example in this paper we use the task of giving a presentation inside our smart meeting room. This environment is introduced below. The graphical representation of this task is given in Figure 1. Here the task of giving a presentation decomposes to a sequence of sub-tasks. The user starts the presentation with entering the room and moving to the front of the room. When the presentation is finished the user moves to the door to leave the room. A more detailed description of this example is given in section 3.

2. PRELIMINARIES

The controller described below is based on semantic models of the user and its environment. For our system we currently employ formal action descriptions and task models which are compiled into a probabilistic model. All necessary concepts are briefly introduced below.

Hidden Markov Models (HMMs) [7] are probabilistic models, that allow to infer a state of a system that is not observable directly, but through noisy or ambiguous sensor data. An HMM defines a probabilistic model, that consists of a finite number of states, each containing a probability distribution function over sensor observation, that allow to conclude the system state given sensor data. To describe temporal behavior of a system an HMM specifies probabilities for state transitions. HMMs are state of the art methods for activity recognition.

Depending on the available sensors, we can detect the current activity of users. For example, an indoor positioning system can be used to detect whether a user is entering the room and heading for the presentation stage. As customary in activity and intention recognition, we use probabilistic models. Such models can cope with noisy and contradictory sensor data and allow nonetheless to infer the most likely sequence of actions or complex intention. Here, we use Dynamic Bayesian Networks [6], such as HMM's for prob-
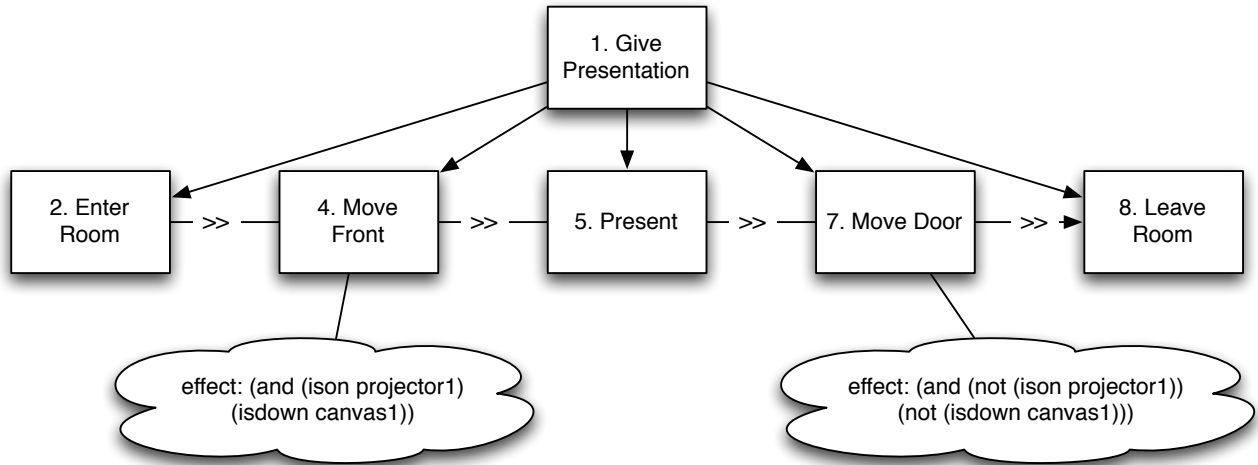
Figure 1: Simplified CTML model describing a typical presentation within our smart environment

abilistic modeling. Calculating a probability distribution over the current state with respect to the observed sensor data as well as the previous state is known as filtering. Doing this requires a model, that describes both, the behavior of the user and the sensor data observable. In addition of recognizing the activity these methods allow to predict future activities, in this case intentions, of the user.

Complex behaviors of (groups of) users can formally be described using CTTE [5] or CTML-models [9], which basically are a hierarchical description of tasks. Sub-tasks can be set into a temporal relation of each other. CTML utilizes temporal operators as the sequence operator (»), the order independence operator (|=|), the concurrent operators (|||) and others that are not used with the examples in this paper. The Collaborative Task Modeling Language (CTML) is especially designed for smart environments and offers features for team modeling, location modeling, device modeling and domain modeling. As described in section 4, we can transfer such a description into a probabilistic model allowing to recognize the current complex action, and thus allowing to infer the overall intention of a sequence of actions.

In the planning domain definition language (PDDL) [8], device actions are formalized as 4-tuples: ⟨Name, Parameters, Preconditions, Effects⟩. Based on such a formal description we can use standard AI planning techniques to infer a plan (sequence of actions) leading from the current to the desired state of the world. Figure 2 and 3 show examples for PDDL descriptions.

3. AN APPLICATION EXAMPLE

The environment where most of our experiments take place is the so called *Smart Appliance Lab*. This room is instrumented with various sensors such as the location tracking system Ubisense [1]. Thus, the location of different users is given by the environment. Other parts of our experimental environment are actuators such as projectors and canvases. Here both sensors and actuators are called devices. Software counterparts of all these devices are provided by the middleware implemented for this environment. These software devices enable us to gain the status of each device inside the room to create a world state. The world state of our environment is thus comprised of the sensor observations and the device states. The

Projector1 on Canvas1 down	true	false
true	TT	TF
false	FT	FF

Table 1: The cartesian product of all device states forming the world state.

environment as well as the middleware controlling the devices of the environment are described in [3]

Since the experimental environment may be used as smart meeting room, a typical application is *giving a presentation*. In this scenario the user first enters the room. For our example we assume that the room contains one projector and one canvas. After the user moves to the front of the room where the canvas is located, he prepares the environment for his presentation. Therefore he has to plug in the notebook, set up the projector and lower the canvas. After this is done the user starts his talk and finishes it by moving to the door. Finally the user leaves the room. This example is kept simple to illustrate the main points. The real environment is comprised of eight projectors and eight canvases.

The task specification in Figure 1 contains a detailed description of this example. The annotated effects (illustrated as clouds) describe the desired state of the environment for the following sub-tasks. As description language for task models we use CTML, as described in section 2. The graphical representation of the task model omits the description of the observation data and the priority function. The world state in this example is given in Table 1 and only consists of the two devices. Each of them has a binary state, in case of the projector it is either turned on or turned off. The canvas can be up or down.

Our goal is now to build a controller that recognizes the current state of the user and executes corresponding device actions that makes the annotated effects come true. By executing these action sequences that system automatically assists the user in achieving his goals.

```
(:action canvasdown
    :parameters (?c – canvas) :
    :precondition (not (isdown ?c))
    :effect (isdown ?c ))
```

Figure 2: A PDDL specification of the CanvasDown action.

```
(:action projectoron
    :parameters (?p – projector) :
    :precondition (not (ison ?p))
    :effect (ison ?p))
```

Figure 3: A PDDL specification of the ProjectorOn action.

4. A CONTEXT-AWARE PROACTIVE CONTROLLER

This section explains how to combine formal descriptions of the environment and the user behavior into a purely reactive probabilistic model. First the description of the user is compiled into a probabilistic model allowing to recognize the user's intentions. Then, we enrich this model by annotating the states with actions, executable by the environment. While running the system, and based on the current state of the environment one of the states will be most likely. The actions attached to the state are then simply executed, resulting in a system supporting the user while achieving his high-level goals.

In the following sections we discuss two different possibilities of enriching the model with device actions. The first is to generate different HMM states for each possible world state. The latter is to annotate the corresponding state with sequences of device actions for all possible world states. A plan for the current world state is then accessible by taking the world state as key for a lookup table resulting in the corresponding plan.

4.1 From Symbolic to Probabilistic Models

We start with the *annotated task model* from Figure 1, which consists of a task model and effects annotated to sub-tasks. Each effect is a subset of the *world state*. It consists of the cartesian product of all device states. We apply the transformation given in [4]. This is done by parsing the syntax tree of the annotated task model and applying the inference rules for each of the temporal operators. The states of the resulting *annotated HMM* correspond to tasks of the task model with corresponding effects. The whole model captures all possible (with respect to the task model) sequences to complete the root task.

We extend the original task model by annotating tasks with their effects with respect to the world state. The effect of the annotated task should be true after executing the task. This can be done by either the user or the controller. Figure 1 contains the additional effect specification for the *Move Front* and the *Move Door* sub-tasks. In our scenario the effect of the state *Move Front* is that the environment is prepared for the presentation, namely the canvas is down and the projector is on.

To ensure the effects of an annotated HMM state become true, the controller has to execute device actions depending on the current world state. The canvas has to be lowered if it is up but if the projector is already turned on we can omit turning on the projec-
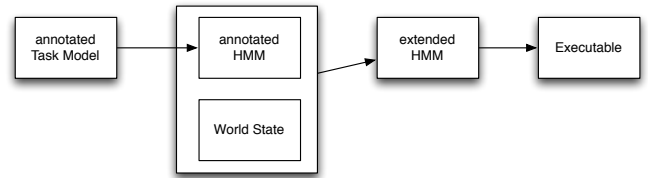

Figure 5: The workflow for creating the controller.

tor. Therefore we have to generate sequences of devices actions for each possible situation. This is done by taking each world state as start situation for a planner and the desired subset of the world state, described by the effects of the annotated HMM state as goal. To realize this the planner takes device action specifications, as shown in Figure 2 and Figure 3. Result of this planning step is a sequence of device actions for each possible world state that has to be executed to create the effects specified in the original annotated task model in Figure 1.

The next two sections describe how to use these world state device action sequence pairs to generate the controller. Both approaches follow the workflow given in Figure 5.

4.2 Unfolding HMM states

In order to create the distinction of the different world states as HMM states, it is necessary to unfold annotated HMM states by using the different world states. Therefore we replace the annotated HMM state by *extended HMM states* that are generated from each possible world state and the state itself. In our example *Move Front* will be replaced by each element of the cartesian product of the world state to ensure that each observation of a world state corresponds to one HMM state. Here *Move Front* is replaced by four new HMM states, each representing a possible world state. Only the HMM state that covers the complete effects has a transition to the HMM state generated from the following sub-task. In our example only the *Move Front TT* state, that assumes that the projector is on and the canvas is down has this transition.

This allows to attach plans to the states in the probabilistic model that needs to be executed in that state as follows: Every annotated user state is combined with every world state, that is a combination of all states of the devices. This world state is used as precondition for device actions during compilation process.

Figure 4 contains the extended HMM that was generated from the task model in Figure 1 combined with the world state defined in Table 1. Please note that self-transitions as well as probabilities for transitions or observations are omitted in the graphical representation.

The generated HMM contains so called slices, that consists of all states generated from one sub-task from the CTML specification. One slice itself was created from the cartesian product of the world state. The intra-slice states differ from each other only by the possible observations of the world state. The names of the intra-slice states illustrated in Figure 4 contain the true/false value given in Table 1. Every state of the slice has an incoming intra-slice transition with a probability given by the number of states. Only the states that create the effects given in task model description have a outgoing inter-slice transition with very high probability. It is possible
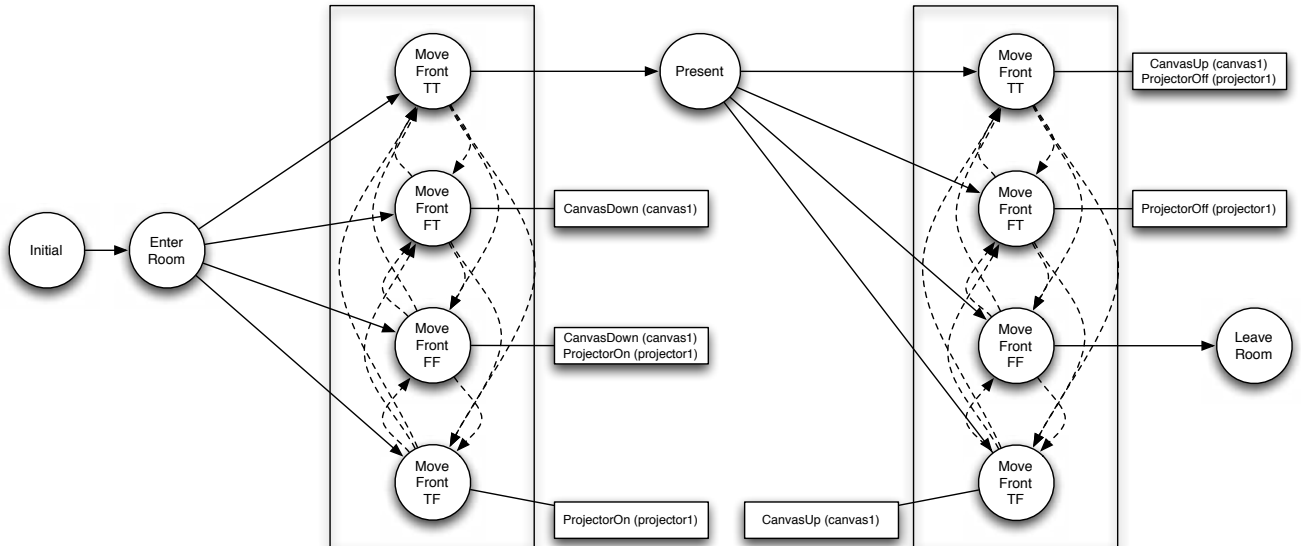
Figure 4: The extended HMM created from the task model and the world state.

that there are more than one state that covers the same effect due to missing effects to single devices of the world. The probabilities of the intra-slice transitions are just given by the number of target states. Inter-slice transitions are generated with respect to the temporal operator of the sub-tasks. The probability of transitions are generated by the normalized weight of the single sub-tasks.

4.3 Lookup table

Another approach to enrich the HMM with device action sequences for user assistance is to create a lookup table for necessary device actions and attach it to the corresponding annotated HMM state. As in the first approach the device action sequences depend on the current world state and need to be generated by a planner that uses the specified effects as goals. Attaching this device action sequence - world state pairs to the annotated HMM state provides a lookup table at runtime. Using the world state, consisting of all device states the table provides the pre-generated device action sequence that has to be executed in order to make the specified effects to the environment become true.

In our scenario the states *Move Front* and *Move Door* are extended by lookup tables for user assistance. The table annotated to the *Move Front* state contains device action sequences that ensure that the projector is turned on and the canvas is down. The *Move Door* HMM state is annotated with a table of device action sequences that ensure that given any world state the projector is turned off and the canvas is up. Figure 6 contains a graphical representation of the generated extended HMM. Probability distribution functions as well as state transition probabilities are omitted for reasons of clarity.

4.4 Choosing one method

The previous sections describe two approaches to attach pre-generated plans to states of a probabilistic model, namely an HMM. Both approaches generate realtime capable systems, that do not have to solve planning problems such as finding a sequence of device actions to support the user. The first approach creates an HMM with
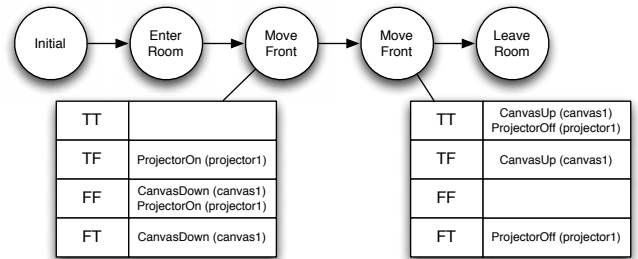
Figure 6: The generated HMM extended by lookup tables for the plans.

very much states, because it creates states for every possible world state. Here the distinction of the world state is done at the state level. The idea of unfolding HMM states by using every possible world state is appropriate if the state of a device is not reliable or noisy.

The second approach is to move the distinction of the world states from different HMM states with attached plans to one HMM state that contains a table of multiple plans, one for each possible world state. The number of HMM states is independent from the world state, which avoids a very high number of states. This approach is applicable whenever the world state is known definitely. This means that each device state is observable without any noise or inconsistency.

5. THE EXECUTION ENVIRONMENT

We developed an execution framework for Bayesian inference that is able to perform fast online filtering of HMM's and particle filters. By separating the model description from the implementation of the algorithms we designed a highly reusable framework. This framework enables users to embed parameterized filters into different environments. This enables us to integrate the generated con-

troller into the software structure described above. Users of this environment only need to implement problem specific details.

A probabilistic model for filtering in our framework has to be implemented in C++. One has to describe three different parts. First a specification of the state space, that is in the case of an HMM represented by a set of states. Second the transition probabilities from each state to another, represented as matrix of probabilities. Third a probability distribution of sensor observations for each state. To provide a more intuitive tool for describing HMM's we introduced a description language that supports a simple way to describe HMM based models.

The compilation process creates the state space of our controller from the single sub-tasks of the user model combined with each possible device state combination. The transition probabilities are given by the probabilities of the model generated only by the sub-tasks as described in [4] and the observation probability distributions of the original approach are extended by an observation of the device states in the extended state. A model specified in the way needs sensor data and the world state as input and provides a sequence of device actions as output. These device actions need to be executed in order to support the user.

6. SUMMARY AND OPEN PROBLEMS

In this paper we showed that a context-aware controller for smart environments can be created from a combination of semantic models of the user and the environment. The controller is based on bayesian inference where the model was generated from task-based specification of users together with a precondition and effect specifications of each device forming the environment. Sensor data as well a accumulated world state serve as input, a device action sequence, that needs to be executed as output of the inference process. We introduced two ideas to merge task based user models and precondition and effects specification of the environment to create probabilistic models that assist the user.

Further research should include smart environment evaluation of the controller described here. Both approaches should be evaluated and the results should be compared for different devices and scenarios. This includes tests for maximum manageable complexity of the state space as well as minimal complexity that creates sufficient user support. Due to the compile time planning process we are able to pre-generate action sequences. This allows to find modeling problems such as deadlocks at compile time. Our approach in this paper utilizes HMM's for inference. However, since the state space may explode and exact inference will not be suitable, we can change the inference algorithm to Monte Carlo based methods such as particle filters. These methods are already supported by the execution environment described above.

The controller introduced here is described as central service. It is possible to decentralize this approach to the usage of multiple services, each of them describing a subspace of the model. By comparing the likelihood of multiple services, it is be possible to choose either an action sequence of one agent or a combination of multiple sequences that do not disturb each other.

Another point that should be analyzed is how both systems behave if only the most probable device action sequences will be pre-planned. If the system reaches a state that does not contain a device action sequence the plan has to created at runtime. The realtime behavior of this extension has to be examined.

Acknowledgements

7. REFERENCES

[1] http://www.ubisense.de, JUN 2009.

[2] *Ubicomp '10: Proceedings of the 12th ACM international conference on Ubiquitous computing*, New York, NY, USA, 2010. ACM. 608109.

[3] S. Bader, G. Ruscher, and T. Kirste. Decoupling smart environments. In S. Bader, T. Kirste, W. G. Griswold, and A. Martens, editors, *Proceedings of PerEd2010*, Copenhagen, SEP 2010.

[4] C. Burghardt, M. Wurdel, S. Bader, G. Ruscher, and T. Kirste. Synthesising generative probabilistic models for high-level activity recognition. In *Activity Recognition in Pervasive Intelligent Environments*. Atlantis Press, Paris, France, 2010. To appear.

[5] G. Mori, F. Paterno, and C. Santoro. Ctte: Support for developing and analyzing task models for interactive system design. *IEEE Transactions on Software Engineering*, 28:797–813, 2002.

[6] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, CA, USA, 2002.

[7] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[8] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2009.

[9] M. Wurdel, D. Sinnig, and P. Forbrig. CTML: Domain and Task Modeling for Collaborative Environments. *J. UCS*, 14(19):3188–3201, 2008.

Towards effective collaborative design and engineering

Stephan Lukosch
Delft University of Technology
Faculty of Technology, Policy, and Management
Jaffalaan 5, 2628 BX Delft, The Netherlands
s.g.lukosch@tudelft.nl

Gwendolyn Kolfschoten
Delft University of Technology
Faculty of Technology, Policy, and Management
Jaffalaan 5, 2628 BX Delft, The Netherlands
g.l.kolfschoten@tudelft.nl

ABSTRACT

Effective collaborative design and engineering has to deal with various challenges. It is essential to create a shared understanding and facilitate interaction in such a way that effective collaboration becomes possible. Free riding, group think or hidden agendas need to be addressed by rarely available process facilitators. Available tools are not regularly used, are not intuitive and often are difficult to adapt to the changing group needs. In order to tackle the above issues, we want to enable effective collaborative design and engineering by offering intelligent collaboration support that supports facilitators of collaboration processes when monitoring collaboration processes and planning process interventions or tool adaptations.

Categories and Subject Descriptors

H.4.1 [**Office Automation**]: Groupware; H.5.3 [**Group and Organization Interfaces**]: Computer-supported cooperative work; K.4.3 [**Organizational Impacts**]: Computer-supported collaborative work

General Terms

Design, Human Factors

Keywords

Collaboration support systems, intelligent collaboration support, facilitation, group support systems

1. INTRODUCTION

Collaboration has become a critical skill as products and services are becoming increasingly complex, no individual has the skills to design, develop and deliver these alone. Collaboration is however, not without challenges. On a group level, it is essential to create a shared understanding, define rules for decision-making and facilitate interaction in such a way that effective collaboration becomes possible [13]. On a process level, free riding, dominance, group think, hidden agendas, are but a few phenomena in group work that make it a non straight-forward effort [16].

Groups might not be able to overcome the challenges of collaboration by themselves [16]. Even if groups are able to accomplish their goals, they can often collaborate more efficiently and effectively using collaboration support [6]. Collaboration support can be comprised by tools, processes and services that support groups in their joint effort. In knowledge oriented organizations, there is often a need or demand for collaboration support. However, tools and technology for group support exist in a variety of shapes from complex computer systems, as e.g. Group Support Systems (GSS), to simple boxes with cards and pencils. Each of these tools can be used by the group to be more successful in sharing ideas and indicating relations and preferences, but current challenges emerge from the fact that available tools are not regularly used, are not intuitive and often are difficult to adapt to the changing group needs [7]. This makes it difficult for organizations to provide their teams with a suitable and adaptable collaboration support that help them accomplish their goals efficiently and effectively.

As discussed in [7], current collaboration support systems focus adaptations with a limited scope. They are either restricted to specific domains or to specific aspects of collaborative work, often focusing on awareness or knowledge management. Compared to this, we aim to create intelligent collaboration support that creates a shared understanding, facilitates collaborative actions across various geographic, temporal, disciplinary, and cultural boundaries and provides intuitive and adaptive tool support. This will allow us to offer collaboration support for a variety of collaborative tasks in a way that groups can use it for themselves without the need for extensive training or a professional facilitator. In this paper, we will as a first step propose a conceptual framework towards intelligent collaboration support. We modeled collaboration processes and identified factors suggesting process changes as well as adaptations. These factors are the basis for this framework of intelligent collaboration support, which will offer us a first step in monitoring groups and predicting the need for facilitation interventions.

In the next section we will explain in detail how facilitators guide collaboration processes. This will lead to a conceptual framework of collaboration support interventions, presented in section 3. Next we will present how this framework can be used to identify specific collaboration situations to create intelligent collaboration support. We will then reflect on this design and end with conclusions and a research agenda.

2. FACILITATING COLLABORATION PROCESSES

One of way of supporting groups in achieving their goals more efficiently and effectively is to support the group by structuring and

guiding their activities. This skill and profession is called facilitation. The facilitation task is described extensively in GSS literature [1, 8, 14]. The task of a facilitator requires both experience and extensive knowledge of group dynamics and facilitation methods. This tasks involves for instance management of the activities the group is performing, quality of their deliverables, relations between the participants and the use of resources and time [9]. This type of process guidance is often offered by someone external to the group, to ensure impartiality and objectivity.

In an effort to reduce the need for professional facilitators, researchers have been coding facilitation practices to enable the separation of the design task of a facilitator and the execution task [11]. In this way a master facilitator called collaboration engineer, can design and transfer a collaborative work practice to practitioners to execute it for them selves based on a short training. This approach is called Collaboration Engineering [3]. To ensure the predictability and transferability of the collaborative work practice, they are designed with design patterns called thinkLets [4].

To realize an intention by means of intervention, two types of interventions are required [2]. First, there are static interventions in which one or more commands are given to initiate the key activities of a process. We will refer to this kind of communication as an instruction intervention. Second, there are dynamic interventions intended to adjust the actions performed by the group to resolve a discrepancy between the facilitator's intentions and the groups' actions. These interventions depend on emergent conditions. We will call these messages adjustment interventions.

The conceptual design of a thinkLet exists of a set of instruction and adjustment interventions described as rules [4]. These rules are similar to rules mimicking human behavior in avatars [2]. Each rule describes for a role an action that needs to be performed using a capability under some set of constraints to restrict those actions. Further, some thinkLets include conditional rules for frequently-required adjustment interventions because specific discrepancies manifest predictably during the execution of an activity based on the thinkLet.

An example of a set of rules are captured in the LEAFHOPPER thinkLet [4]:

1. Allow participants to add in parallel any number of contributions to any category.

2. Allow participants to add only contributions that are relevant to the categories in which they are placed.

3. Allow participants to add only contributions that match to the contribution specification.

4. Let participants shift focus from category to category as interest and inspiration dictate.

5. Ensure that participants read the contributions of others for inspiration.

3. CONCEPTUAL FRAMEWORK

In order to provide intelligent collaboration support, we first need to identify the key goals that guide facilitation interventions. When facilitators intervene to initiate activity they can offer these at different levels [15]:

1. **Collaboration process design**: Interventions to guide collaborators in choosing appropriate tools and techniques to support the collaboration process.

2. **Collaboration process execution**: guidance to move from one activity to a next activity, changing the collaboration support environment to transfer between activities, while taking documents and decisions along to a next phase.

3. **Collaboration process guidance**: Activities need to be initiated and guarded to execute the collaborative activity.

4. **Collaborative behavior guidance**: guidance in determining and adjusting improves collaborative effectiveness.

In a face to face context, facilitators can make adjustment interventions based on behavior of group members, including communication with group members, quality of the output of the group, and progress versus planned time for the group task. Based on our experience and discussion with expert facilitators, the following list is a first attempt to identify factors used to determine the need to make an intervention:

- **Group**: behavior, emotions, communication, body language, address facilitator, gestures

- **Task**: amount of input, rate of input, quality of input, quality of output, shared understanding, fit in relations in output

- **Time**: progress, time left

Some of these aspects are non-digital and based partially on interpretations. This requires a *translation* to gain the same insights from the online interaction. For instance facilitators might monitor de-focus of participants as an indicator that the group is finished with the task. However, this might also be learned from a significant decrease in input rate. However, without technology support to monitor input rate, perhaps per participant, this would be difficult to detect for a facilitator. Also the interpretation of input rate requires some experience and understanding of the cognitive implications of tools and knowledge sharing. Therefore we need more than a *thermometer* to measure input rate, we need an intelligent collaboration support system that can monitor these factors, and use them to reason about the current collaboration process in order to support facilitators in making intervention decisions.

4. LEAFHOPPER FACILITATION INTERVENTIONS

In the textbox below we describe what a facilitator does after initiating the LEAFHOPPER thinkLet to brainstorm ideas in categories. Underlined are those indicators the facilitator uses to make decisions on interventions. Some of these indicators can directly be observed, others are an interpretation of the facilitator.

After initiating the Leafhopper the facilitator needs to maintain several rules. The contributions of the group need to meet the quality intended, they need to meet the contribution specification, the category in which they are placed. The contributions need to be made in a certain timeframe, and they need to cover a certain scope of information (completeness). Additionally the facilitator will need to maintain a safe and respectful atmosphere to ensure that people feel free and encouraged to participate. To ensure that the participants can share all relevant contributions, the facilitator can add
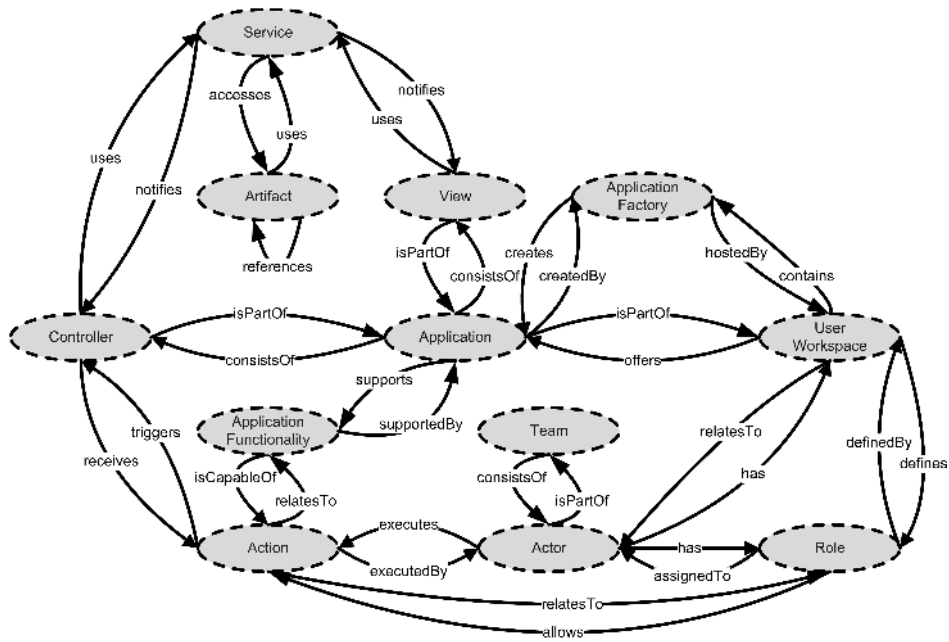
Figure 1: Domain model for collaboration in a shared workspace

a category 'other'. This category is monitored by the facilitator. When a pattern of contributions can be found in this category, the facilitator will add a new category to cover this topic.

The facilitator will monitor the input, mainly to detect if there are **small or insufficient quality contributions**. Later in the process the facilitator will monitor if the categories each contain a **sufficient number of contributions**. Also, the facilitator will monitor the 'other' category to **see if there is a persistent topic addressed**, and therefore, a need to add a category. The facilitator might intervene if some **categories are not filled**. Such intervention would be made **before the time for the task is passed**, to give participants time to add ideas in these categories, but not **too early**, when participants might not yet had a chance to contribute to all categories. The facilitator will also observe the group to see if **participants get distracted**, or **focus on other activities**, which indicate that they are (no longer) motivated for the task. The facilitator will also monitor behavior, communication and body language to see if any of the input causes an **emotional reaction**, which could indicate conflict or flaming, which would require intervention. Finally the facilitator will monitor the **input rate** and the **focus of participants** to detect when there is no more inspiration and the task can be ended. If the group is still very **active** and **focused** when **time is running out**, the facilitator might encourage the group to speed up or to focus on more important contributions in order to ensure that **sufficient progress** is made when the task should be finished. In some cases this can also be a reason to give the group more time for the task.

5. DESIGNING INTELLIGENT COLLABORATION SUPPORT

In order to create a collaboration support system that can suggest facilitators to make interventions, we use an explicit context model to describe the current collaboration situation. A collaboration situation can be characterized by the configuration of the collaboration environment as well as the state of interaction of the users with the system (e.g., based on interaction history) and the organizational

setting (e.g., team structure, roles, tasks). Dey et al. [5]define context as any information used to characterize a situation of an entity where an entity may be any object, person or place providing information about the interaction between a user and an application. With this definition, any information may help characterizing the situation of the interaction's participants because it is part of the context itself. For our purposes, we can narrow this definition so that context includes all information which is necessary or helpful to adapt a shared workspace to better fit the needs of a collaborating team. This implies that the context contains information about the team as well as about the current collaboration situation. This context information is necessary to recognize situations which demand a facilitator's intervention (and thus help minimizing the effort needed for adaptation).

We use a collaboration domain model for describing collaboration environments and collaboration situations [7]. Figure 1 summarizes this domain model and shows the basic classes and their relations that can be used to describe collaboration context in a global collaboration space. The domain model intends to capture the basic concepts of collaborative workspaces. It focuses on the technological support for collaborative interaction and does not distinguish different artifact types or task domains. If applied to a certain collaboration environment, it must be extended with concepts matching the specific properties.

The model in Figure 1 distinguishes different concepts that describe collaboration in a collaboration environment and relations between these concepts. We start exploring and explaining the model in Figure 1 with the concept of an *Actor* (see lower part of Figure 1). The domain model assumes that *Actors* are member of a *Team* and have a *Role* defined by the *User Workspace*, as *Applications* are started from within the *User Workspace* and thus the workspace can ensure pre-defined *Roles*. Each *Role* allows an *Actor* to perform specific *Actions*. The available *Actions* are defined by the supported *Application Functionality* of an *Application*. As an example con-

sider a chat application which should offer at least two action types: *OpenChat* and *SendMsg*. These two actions would allow users to communicate with each other by opening a chat tool and send messages to each other. Other forms of collaboration such as within a collaborative diagram editor would require to add additional action types in order to specify the application functionality.

As *Actors* interact with the *Application* by performing *Actions* allowed by their *Roles*, *Roles* define interaction possibilities within an application, e.g. in a shared writing application an author might perform all edit actions whereas a reviewer can only comment existing text. The *Actions* are received by the corresponding *Controller* components of the *Application*. An *Application* implements the model-view-controller (MVC) paradigm [12] and consists of *Views* and *Controllers* components. *Views* and *Controllers* use *Services* to access the *Artifacts*. *Artifacts* use *Services* to notify *Views* and *Controllers* about changes. Each *Application* is part of a *User Workspace* and is created by an *Application Factory* which specifies what *Applications* are available within a workspace and how these can be initialized. Finally, the class *Application Functionality* specifies the functionality an *Application* offers, e.g. in relation to communication, shared editing, or awareness.

All above classes are useful to model and store the configuration of a collaboration environment and to capture the current context at runtime. Based on such context information, a collaboration environment is enabled to recognize situations, which demand a facilitator's attention and intervention.

The domain model is abstract and not related to a specific application domain. When considering our example on the LEAFHOPPER thinkLet, we need to extend the model as shown in Figure 2. In order to incorporate the LEAFHOPPER thinkLet, the *Artifact* class, the *Action* class and the *Role* class were extended. Based on this extension, we can now distinguish between participants and the facilitator as well as identify contributions within a category.

Based on the extended domain model, we can suggest process interventions or tool adaptations in order to improve collaborative interaction. One process intervention within our LEAFHOPPER example is triggered when the category 'other' exceeds a specified threshold. The following rule consists of a condition and an action block. The condition block retrieves all contributions within the context model that belong to the category 'other' and then evaluates whether the number of contribution has exceeded a specified threshold. If this is the case, the action block opens an alert view for the facilitator. The following pseudo code shows how such a rule can be specified:

```
rule "create new category"
when
  $contributions: Contribution(category ==
    'other')
  eval($contributions.size() >= 20)
 then
  openForFacilitator(Alert, "Number of
                     contributions in
                     category 'other' has
                     exceeded specified
                     limit. Check whether
                     new category is
                     necessary.")
end
```

Another example for a rule that monitors whether there are empty

categories and in case again alerts the facilitator can specified as follows:

```
rule "empty categories"
when
  $category: Category(size == 0)
then
  forall $c in $category
    openForFacilitator(Alert, "Category "+
                        $c.name()+" is empty.
                        Focus the attention on
                        the participants on the
                        empty category.")
end
```

As final example, the following rule checks the focus of the participants in order to alert the facilitator when half of the participants do not focus on the activity of creating contributions. For that purpose, the rule retrieves for focus of each participant by identifying the active *View* in the *User Workspace*. Based on the basic collaboration model (cf. Figure 1), this information can be inferred via the *User Workspace* and the opened *Applications* within the workspace. The following example rule assumes that the participants should focus on a view with the name 'contribution input' and if they do not do so alerts the facilitator:

```
rule "participants distracted"
when
  $participants: Participant(focus !=
             "contribution input")
  $threshold: $participants[0].team().
             size()/2
  eval($participants.size() >= $threshold)
then
  openForFacilitator(Alert, "More than 50%
                     of the participants do
                     not focus on creating
                     contributions.")
end
```

6. DISCUSSION AND CONCLUSION

Collaboration has become a critical success factor for many organizations, as products and services are becoming increasingly complex and cannot be designed individually. However, collaboration has several challenges. It is essential to create a shared understanding and facilitate interaction in such a way that effective collaboration becomes possible. Free riding, group think or hidden agendas need to be addressed by rarely available process facilitators. Available tools are not regularly used, are not intuitive and often are difficult to adapt to the changing group needs. In order to tackle the above issues, we want to enable effective collaborative design and engineering by offering intelligent collaboration support that supports facilitators of collaboration processes when monitoring collaboration processes and planning process interventions or tool adaptations.

In this article, we identified several factors that are observed by professional facilitators before changing and adapting an ongoing collaboration process. We further introduced an abstract context model which can be used to model collaboration within a shared workspace. We extended this model to include concepts and classes of the LEAFHOPPER thinkLet. Based on the experiences of a professional facilitator, we used this extended context model to define rules which can assist a facilitator.

Based on the proposed rules, a context-adaptive and intelligent collaboration support environment, such as [17], can alert a facilita-
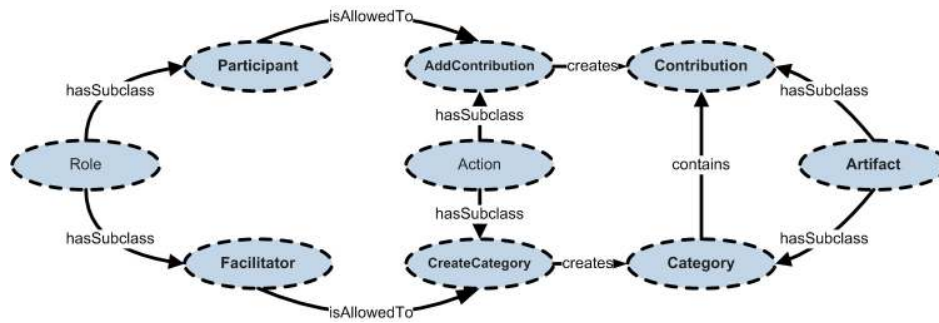
Figure 2: Extended domain model for the LEAFHOPPER **thinkLet**

tor when an intervention might become necessary and reduce the facilitator's overhead. In future work, we will go a step further and model entire collaboration processes based on thinkLets [10]. We then will study factors that determine and influence collaboration performance, e.g. cognitive load or shared understanding, and that inform facilitation interventions. Once identified, we will integrate these factors in our context model and think about possibilities to measure soft factors via additional application functionality and without obstructing or distracting the group work, e.g. by offering means for self reporting. We will further identify and specify rules that recognize situations that require process interventions by recording facilitation interventions and the performance indicators at the point of intervention to gain more fine-grained rules for process intervention.

The path to intelligent collaboration support sketched above is long, but small steps might already improve collaborative design and engineering today. As facilitators need to monitor many factors and indicators of progress, basic suggestions for process intervention as outlined above might already reduce some of the cognitive load of the facilitation task.

7. REFERENCES

[1] F. Ackermann. Participants perceptions on the role of facilitators using group decision support systems. *Group Decision and Negotiation*, 5:93–519, 1996.

[2] N. Badler, R. Bindiganavale, J. Bourne, M. Palmer, J. Shi, and W. Schuler. A parameterized action representation for virtual human agents. In *Workshop on Embodied Conversational Characters*, 1998.

[3] R. Briggs, G. de Vreede, and J. Nunamaker. Collaboration engineering with thinklets to pursue sustained success with group support systems. *Journal of Management Information Systems*, 19:31–63, 2003.

[4] R. Briggs and G.-J. de Vreede. *ThinkLets: Building Blocks for Concerted Collaboration*. Delft University of Technology, Delft, The Netherlands, 2001.

[5] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2, 3, & 4):97–166, 2001.

[6] J. Fjermestad and S. Hiltz. A descriptive evaluation of group support systems case and field studies. *ournal of Management Information Systems*, 17:115–159, 2001.

[7] J. M. Haake, T. Hussein, B. Joop, S. Lukosch, D. Veiel, and J. Ziegler. Modeling and exploiting context for adaptive collaboration. *International Journal for Cooperative Information Systems (IJCIS)*, 19(1-2):71–120, 2010.

[8] S. Hayne. The facilitator's perspective on meetings and implications for group support systems design. *Database*, 30(3-4):72–91, 1999.

[9] G. Kolfschoten and G. de Vreede. A design approach for collaboration processes: A multi-method design science study in collaboration engineering. *Journal of Management Information Systems*, 26:225–256, 2009.

[10] G. Kolfschoten, S. Lukosch, and M. Seck. Modeling collaboration processes to understand and predict group performance. In A. Dix, T. Hussein, S. Lukosch, and J. Ziegler, editors, *Proceedings of the IUI workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS) 2010*, 2010.

[11] G. Kolfschoten, F. Niederman, G. de Vreede, and R. Briggs. Roles in collaboration support and the effect on sustained collaboration support. In *Hawaii International Conference on System Science (HICSS-41)*, 2008.

[12] G. E. Krasner and S. T. Pope. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49, Aug. 1988.

[13] S.-Y. Lu, W. Elmaraghy, G. Schuh, and R. Wilhelm. A scientific foundation of collaborative engineering. *CIRP Annals - Manufacturing Technology*, 56(2):605 – 634, 2007.

[14] F. Niederman, C. Beise, and P. Beranek. Issues and concerns about computer-supported meetings: The facilitator's perspective. *Management Information Systems Quarterly*, 20(1):1–22, 1996.

[15] F. Niederman, G. de Vreede, R. Briggs, and G. Kolfschoten. Extending the contextual and organizational elements of adaptive structuration theory in GSS research. *Journal of the Association for Information Systems*, 9(10), 2008.

[16] J. J. Nunamaker, R. Briggs, D. Mittleman, D. Vogel, and P. Balthazard. Lessons from a dozen years of group support systems research: A discussion of lab and field findings. *Journal of Management Information Systems*, 13:163–207, 1997.

[17] D. Veiel, J. M. Haake, and S. Lukosch. Facilitating team-based adaptation of shared workspaces. In *International Symposium on Collaborative Technologies and Systems (CTS 2010)*, pages 275–284. IEEE, 2010.

Graphs of models for exploring design spaces in the engineering of Human Computer Interaction

Alexandre Demeure, Dimitri Masson
Laboratory of Informatics of Grenoble
655 Avenue de l'Europe
38330 Montbonnot-Saint-Martin, France
+33 (0)4 76 51 48 54
firstname.lastname@inrialpes.fr

Gaelle Calvary
Laboratory of Informatics of Grenoble
385, rue de la Bibliothèque - B.P. 53 - 38041
Grenoble Cedex 9, France
+33 (0)4 76 51 48 54
gaelle.calvary@imag.fr

ABSTRACT

Model Driven Engineering (MDE) has focused on the latest stages of the design process so far and as a result has missed the opportunity to foster creativity in the early phases. Our research aims at stretching MDE all over the design process including the creative phases so that to go beyond the well-known 'fast-food UIs' limit of MDE. We propose to consider sketches and prototypes as models. This paper claims for storing these models in a graph so that to both inspire designers and support adaptation at runtime.

Keywords

Model based User Interfaces, graph of models, design spaces, creativity.

ACM Classification Keywords

H.5.2 [**Information interfaces and presentation**]: User interfaces – *prototyping.*

INTRODUCTION

Early phases of User Interfaces (UI) design require the production of numerous propositions so that to result in a successful design [1, 11]. Those propositions are usually explored through sketches and prototypes that quickly materialize designers' ideas as a support for discussion, selection and validation. Whilst those early phases are crucial for good design, we observe that currently Model Driven Engineering (MDE) sustains the latest stages of design only (i.e., when the code of the concrete UI is produced). This can be explained by the historical grounding of MDE that comes from software engineering. Those approaches aim at proposing optimal solutions for a given problem in a particular context (e.g. SUPPLE [5]) but not at sustaining human creativity. As a result, MDE seems to be pushed at its limits [2]: advanced UIs or aesthetic UIs seem to be out of range.

We believe that the relative disappointment with regard to MDE is due to this lack of support of early phases. In this paper, we propose to consider sketches and prototypes as models to support the exploration of numerous ideas. We store these models in a graph that makes explicit the relationships between models. This graph and the related exploring tools are currently work-in-progress.

RELATED WORKS

Buxton [1] and Tohidi [11] elicit sketching and prototyping as key for creative designs whatever the domain is. Buxton [1] stresses that the value of sketches does not lie in the produced artifact itself (the drawing) but in its ability to trigger the desired and appropriate behaviors, conversations and interactions. Indeed, sketches are a vehicle, not a target: designers do not draw sketches to depict ideas that are well consolidated in their mind. Rather, they draw sketches to try out vague and uncertain ideas. When seeing the sketches, designers can spot problems they may not have anticipated. Even more, they can see new features and relations among elements that they have drawn. Some of them were not intended in the original sketches. These unintended discoveries promote new ideas and refine current ones.

Tools exist to help designers to sketch and prototype UIs. A simple yet quiet efficient example is a pen coupled with a sheet of paper. However, paper based sketches are not really appropriate to describe interaction. In some cases, this shortcoming can simply be overcome by using animated GIF. More generally, electronic tools such as SILK [6] or DENIM [8] have been developed to enable designers to quickly specify the interaction directly from sketches. Other tools such as SketchiXML [3] enable the designers to sketch a UI that is then interpreted as a set of UsiXML widgets. However, the set of widgets is not extensible (i.e. a brand new widget can not be added), which is a strong limitation for creativity.

Demeure [4] explored semantic graphs for storing and reusing UI components both at design time and runtime. Masson [9] investigated genetic algorithms as a support for exploring possible UIs for a given task by assembling UI components that correspond to the (sub)tasks and tasks operators. However, in both cases, the components were formally described. Thus sketches and prototypes were not taken into account which dramatically limits the design space exploration.

STRUCTURE OF THE GRAPH OF MODELS

As in [4], we propose to organize the UIs' models in a graph but enriched with informal models such as sketches and prototypes.

Nodes of the graph

Nodes of the graph are UIs' models defined at one of the CAMELEON levels of abstraction: Concepts and tasks (C&T), Abstract UI (AUI), Concrete UI (CUI) and Final UI (FUI). Each node is enriched with a level of precision. This level ranges from "rough sketch" to "formal definition", covering all levels of fidelity in prototyping.
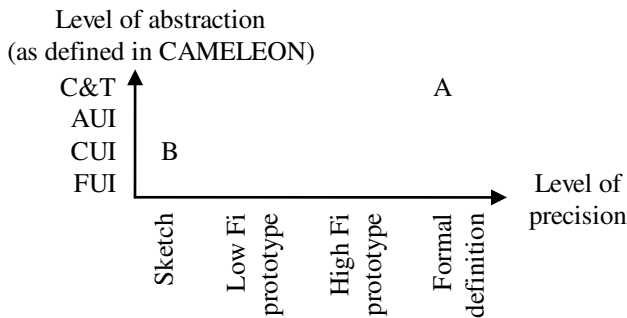

Figure 1: Nodes are characterized by a level of abstraction and a level of precision. A and B are two samples detailed below.
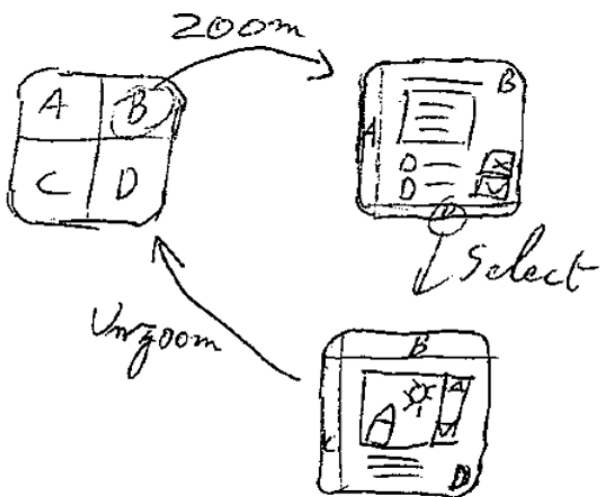

Figure 2: An example of Point B in Figure 1: the node is the interleaving task operator. It is defined at the CUI and Sketch level.

Point A in Figure 1 may correspond to a formal definition of the interleaving task operator. Such a definition could be based on CTT [10]. More concrete descriptions of this operator could be provided. For instance, point B is a concrete description of this operator but at a sketch level of precision only. Figure 2 provides an example of such a CUI-Sketch definition.

Arcs of the graph

The arcs of the graph model the relationships between UI models. Arcs can be seen as transformations that produce target UI models from source UI models. A transformation is defined by:

- A level of precision ranging from informal to formal;

- The context of use (in terms of platform, user and environment) the transformation requires;

- A degree of originality that conveys how much the know-how expressed in the arc is spread over designers: is it shared by the whole HCI community, or just by a part of it? This attribute gives designers clues on how well established or how innovative the transformation is.

Figure 3 illustrates a possible classification of transformations. This classification goes beyond usual transformations that are limited to the levels of abstraction they manipulate (Abstracts and Concretizes). Thanks to our classification, transformations can also be used for:

- Changing the level of precision of UI models (e.g., providing a formally defined UI model from an informal prototype).

- Making the composition of a UI model explicit (e.g., a task tree is composed of subtasks and task operators).

- Expressing that a UI model is another version of another one. This can be useful for knowing that UI alternatives exist.

Overall, transformations are a means for expressing the design rationale of an evolution in the design process.
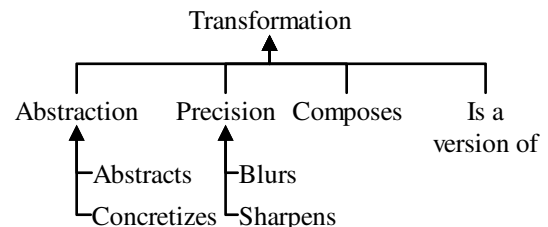

Figure 3: Classification of transformations.

EXPLOITATION OF THE GRAPH OF MODELS

This section develops how powerful the graph is to support evolution both at design time and at runtime. Figure 4 is used to support explanation.

At design time

At design time, the graph serves two purposes: 1) to inspire designers by capitalizing the know-how in UI design, and 2) to provide a space to store and access UIs produced by designers during the design process.

The graph provides a means for designers' teams to structure their production of sketches and prototypes. Relationships between the UI models can embed the design rationale of the design process (the motivations of the design choices). For instance, in Figure 4, a project starts by a sketch of a C&T description. Neither the tasks nor the concepts are well defined, but stakeholders agree on an informal description of the project. Then this description is sharpened to a formal C&T model (here a CTT model). Nodes C, D, E, F and G describe one possible design evolution: from the C&T model, designers explore two paths: C followed by E and G, in parallel with F. C is more thoroughly explored. Several design versions are proposed and explained. The last version (G) sharpens parts of the design.

The graph stores the evolutions, discussions, and choices along with their rationale. Thus designers can later on go back to understand where an idea comes from, or start a new branch while keeping memory of alternatives. Indeed, different parts of the design may evolve at different places in the graph, or along different paths. In a same node, some parts can be highly detailed denoting a high level of confidence in the design choice, whilst other parts can still be roughly sketched (for instance node G in Figure 4 where only a part of the UI is sharpened).

Designers can select parts of a drawing and link them to other nodes, or parts of other nodes. For instance, designers can specify that one part of the C&T model represents the "Manage contacts list" and link it with the corresponding nodes. They can also link it to the circle part in node C. This possibility to identify parts of models is particularly useful when applied together with the "Composes" relationship. Designers can specify that a node is composed of several sub-nodes. In the case of a C&T model, sub nodes may represent sub tasks involved in the model. The "Composes" relationship makes it possible to split problems carried out by models into sub-problems. This is key for reducing complexity by finding, capitalizing and reusing solutions to smaller problems.

Designers can then explore possible solutions by assembling solutions of sub-problems together. As sub-problems can be decomposed in turn, this leads to a combinatory explosion and makes it impossible for designers to explore all of them. Thus one solution is to let the exploration of the combinations to search algorithms. Masson [9] proposed to use genetic algorithms to produce examples of UIs designs. Based on an external database that capitalizes widgets at several levels of abstraction (C&T to FUI), it takes a C&T model in input and produces a set of transformations to be applied on the C&T model to produce final UIs. However this approach focuses on widgets at a very high level of precision only. As a consequence, the generated UIs might not be suitable for early design phases. This approach can be extended to sketches and prototypes.

At runtime

Designers can rely on nodes and arcs at the formal definition level to propose automatic UI generators that can produce UI adapted to a given context of use. Indeed, for a given task, one can go through arcs and nodes to retrieve all possible implementations of this task. For each of these implementations, the path that links it with the original task informs about the context of use it is designed for. For instance, in Figure 4, one can follow the concretization arcs from the interleaving node to find all possible solutions to represent it. This process can be guided by the information about the context of use the node requires. By doing so, it is possible to retrieve all CUI/FUIs adapted to a given context of use. This was explored in [4]. It is related to a service broker devoted to HCI.

The graph, used as a service broker, could be integrated in automatic UI generation algorithms like SUPPLE [5]. The richer the graph is for a given task, higher the chance is to produce adapted UIs. Thus the openness and extendibility of the graph is key compared to closed or non explicit approaches that enumerate possible renderings for tasks or tasks operators. Actually, algorithms like SUPPLE [5] can be seen as a concretization arc in the graph that produces a CUI/FUI (at the formal definition level precision) based on a C&T description (at the formal definition level precision), a user model (his/her UI preferences, Fitts parameters and typical traces) and the targeted platform (widgets set and screen size). Applying SUPPLE to a particular task tree results in adding an arc in the graph starting from the node that embeds the C&T description to a node that describes the generated CUI/FUI. For instance, in Figure 4, SUPPLE can be applied to the C&T node that describes the instant messenger to produce a CUI (B in Figure 4) optimized for the platform P and user characteristics U.

CONCLUSION

Considering sketches and prototypes as models in MDE is promising to avoid the "fast-food UI" limit. It should enable UI designers to take advantages of these powerful approaches while taking benefit of the strong know-how HCI has in MDE.

We explore how capitalizing models in a graph can be useful both at design time and runtime to get inspired and

make the exploration of the design spaces easier. The implementation of this graph and the related exploration tools is currently work-in-progress. We plan to involve UI designers in their design as well.

REFERENCES

1. Buxton, B. *Sketching User Experiences: Getting the Design Right and the Right Design*. 448 pages, Morgan Kaufmann (March 30, 2007). ISBN-13: 978-0123740373.

2. Coutaz, J., *User Interface Plasticity: Model Driven Engineering to the Limit!*, in ACM, Engineering Interactive Computing Systems (EICS 2010) International Conference. Keynote paper. Pages 1-8. 2010.

3. Coyette, A., Faulkner, S., Kolp, M., Limbourg, Q., Vanderdonckt, J., *SketchiXML: Towards a Multi-Agent Design Tool for Sketching User Interfaces Based on UsiXML*, Proc. of 3rd Int. Workshop on Task Models and Diagrams for user interface design TAMODIA'2004 (Prague, November 15-16, 2004), Ph. Palanque, P. Slavik, M. Winckler (eds.), ACM Press, New York, 2004, pp. 75-82.

4. Demeure, A., Calvary, G., Coutaz, J., and Vanderdonckt, J. *The comets inspector: Towards run time plasticity control based on a semantic network*. In TAMODIA'06.

5. Gajos, K., and Weld, D. S. *Supple: automatically generating user interfaces*. In IUI '04: Proceedings of the 9th international conference on Intelligent user interface (New York, NY, USA, 2004), ACM Press, pp. 93–100.

6. Landay, J.A. and Myers, B.A. *Interactive sketching for the early stages of user interface design*. Proceedings of the SIGCHI conference on Human factors in computing systems, 1995.

7. Lee, B. and Srivastava, S. and Kumar, R. and Brafman, R. and Klemmer, S.R. *Designing with interactive example galleries*. Proceedings of the 28th international conference on Human factors in computing systems, 2010, pp. 2257--2266

8. Lin, J. and Newman, M.W. and Hong, J.I. and Landay, J.A. *DENIM: finding a tighter fit between tools and practice for Web site design*. Proceedings of the SIGCHI conference on Human factors in computing systems, 2000.

9. Masson, D. and Demeure, A. and Calvary, G. *Magellan, an Evolutionary System to Foster User Interface Design Creativity*. In proceedings of EICS'10, Berlin, 2010.

10. Paterno, F. and Mancini, C. and Meniconi, S. *ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models*. In Proceedings Interact'97, July'97, Sydney, Chapman&Hall, 1997, pp. 362-369.

11. Tohidi, M., Buxton, W., Baecker, R., and Sellen, A. *Getting the right design and the design right*. In CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems (New York, NY, USA, 2006), ACM, pp. 1243–1252.
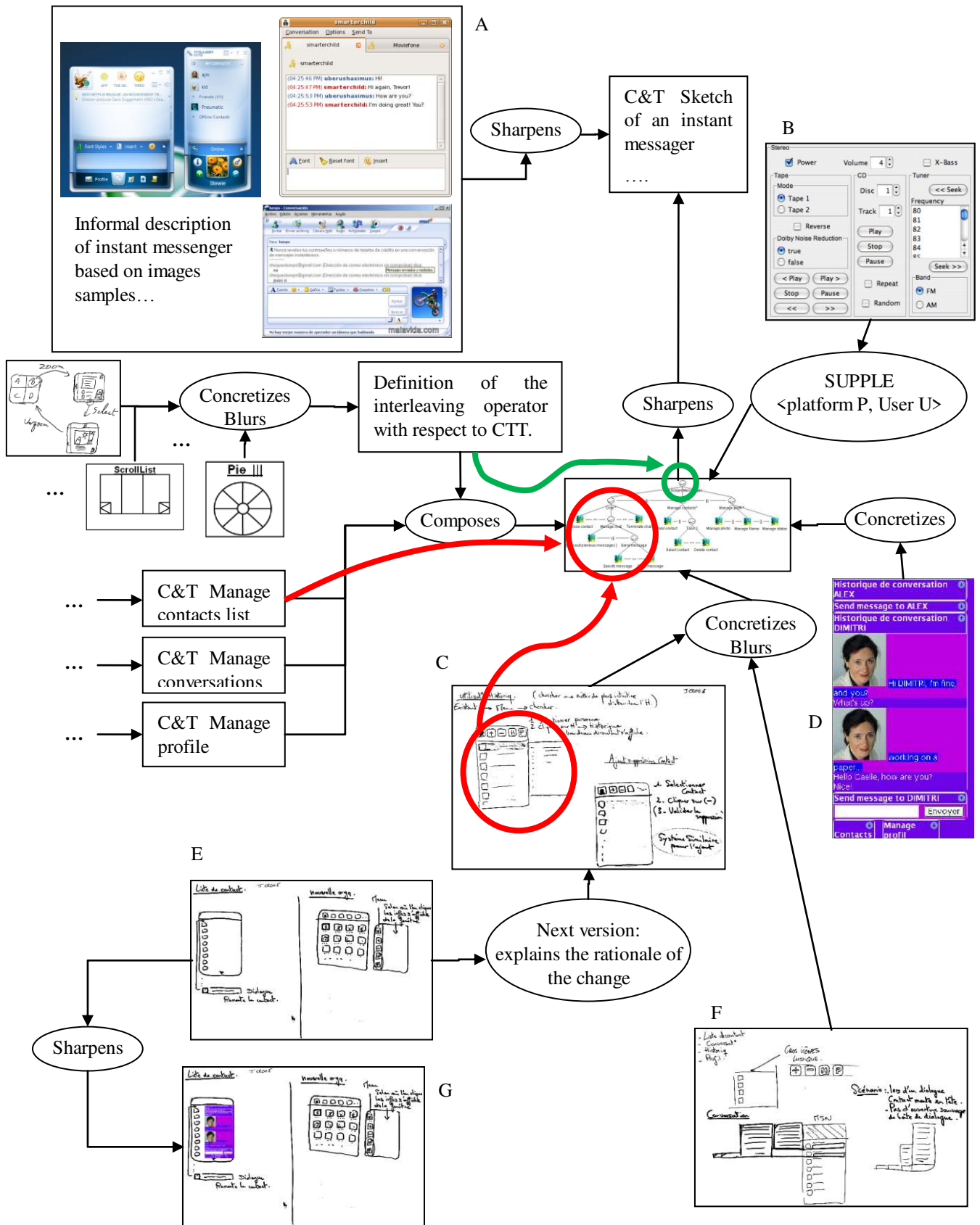
Figure 4: Excerpt of a graph of UI models.

A Formal Ontology on User Interfaces –
Yet Another User Interface Description Language?

Position Paper

Heiko Paulheim and Florian Probst
SAP Research
Bleichstrasse 8
64283 Darmstadt, Germany
{heiko.paulheim,f.probst}@sap.com

ABSTRACT
During the past years, a lot of user interface description languages, most of them based on XML, have been introduced. At the same time, the use of formal ontologies for describing user interfaces has been discussed for a number of use cases. This paper discusses the differences between a formal ontologies and user interface description languages and and points out how both research directions can benefit from each other.

Author Keywords
User Interfaces, Ontology, UI Description Languages, Formal Models

ACM Classification Keywords
D.2.2 Software Engineering: Design Tools and Techniques—*User Interfaces*; D.2.11 Software Engineering: Software Architectures—*Languages*; I.2.4 Artificial Intelligence: Knowledge Representation Formalisms and Methods—*Semantic Networks*

General Terms
Design, Languages

INTRODUCTION
Recently, a number of use cases have been proposed that employ ontologies for modeling user interfaces, their components and interaction capabilities. Examples are automatic generation of explanations for user interfaces, adaptation of user interfaces for different needs and contexts, and integration of user interface components [14]. Those use cases require a strongly formalized ontology of the domain of user interfaces and interactions.

In parallel, various UI description languages have been proposed, most of them XML based [7, 12]. The duality of UI description languages and formal ontologies gives rise to the question whether an additional ontology is really needed, or whether it is going to be yet another user interface description language.

ONTOLOGIES AND MODELS
Although ontologies and software models are related, they are not essentially the same. Software models and ontologies are different by nature. An ontology claims to be a *generic*, commonly agreed upon specification of a conceptualization of a domain [6], with a focus on precisely capturing and formalizing the semantics of terms used in a domain. A software model in turn is *task-specific*, with the focus on an efficient implementation of an application for solving tasks in the modeled domain [2, 16, 18]. Thus, a software engineer would rather trade off precision for a simple, efficient model, with the possibility of code generation, while an ontology engineer would trade off simplicity for a precise representation. Another difference is that in software engineering, models are most often *prescriptive* models, which are used to specify how a system is *supposed* to behave, while ontologies are rather *descriptive* models, which describe how the world *is* [1]. Figure 1 illustrates those differences.

Taking this thought to the domain of user interfaces and interactions, models are used to define particular user interfaces (e.g. with the goal of generating code implementing those interfaces), while a formal ontology would capture the nature of things that exist in the domain, e.g., which types of user interfaces exist, and how they are related.

Due to those differences, we argue that developing a formal ontology on user interfaces will not lead to yet another user interface description language, but to a formal model with different intentions and usages. In the next sections, we will discuss how the two worlds can benefit from each other.

HOW A FORMAL ONTOLOGY CAN BENEFIT
FROM UI DESCRIPTION LANGUAGES
A lot of research work has gone into the development of different user interface description languages. Those research efforts can be and should be taken into account when developing an ontology of the domain.

Collection of Concepts
Most methodologies for ontology engineering foresee the capturing of key concepts and relationships as one of the first steps. This can be done by conducting interviews
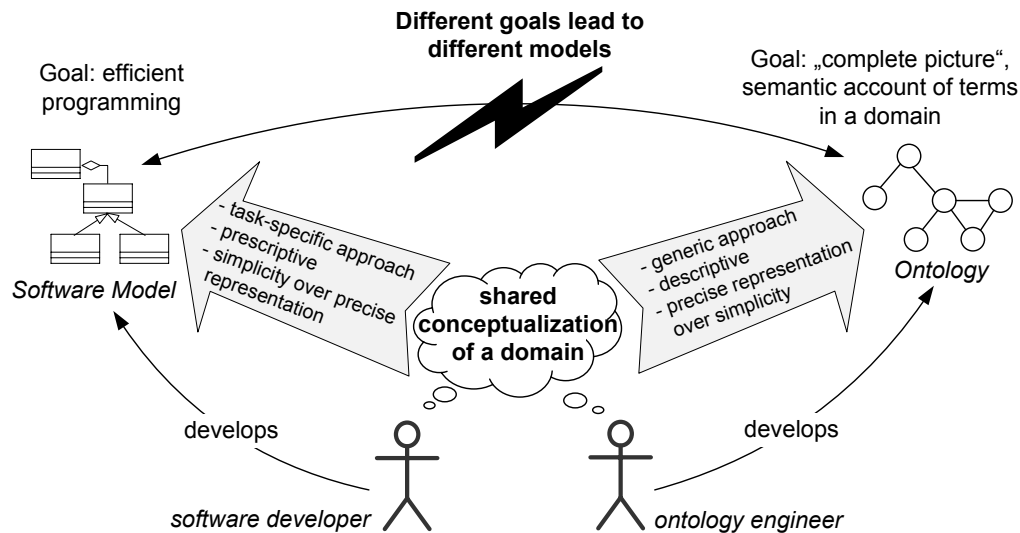
Figure 1. Ontologies and modeling languages serve different purposes.

with domain experts, scanning books and other material, and/or reusing parts of other ontologies [5, 19]. At this point of ontology engineering, lots of input can be used from existing user interface description languages.

Since those languages are most often XML-based, they consist of a smaller or larger number of tags and attributes, which determine the expressivity of the language. As many of those elements define certain concepts of the domain, such as UI components or actions that can be performed with them, they are a good starting point for developing a formal ontology of the domain.

Benchmarking the Ontology's Completeness
As discussed above, ontology engineering aims at providing a complete, comprehensive formal description of a domain. However, assessing the completeness of an ontology is not always an easy task. Here, user interface description languages can once again help by providing a benchmark for the ontology's completeness.

Such a benchmark can be performed in different ways. On the meta-model level, the number of concepts contained in the meta model (e.g., tags and attributes in an XML schema) which have a counterpart in the ontology can be determined. On the model level, one can check whether given models in a user interface description language can be expressed using only the terms given in an ontology, either informally, or formally, e.g., in RDF. Thus, user interface description languages can provide a measure for the completeness of an ontology of the domain.

HOW UI DESCRIPTION LANGUAGES CAN BENEFIT FROM A FORMAL ONTOLOGY
Once an ontology of the domain of user interfaces and interactions has been created, it can be used to improve the development and usage of new and existing user interface description languages as well.

Disambiguation of Terms
In an analysis of user interface description languages, we have found that terms are often used differently in different standards. An example is the term *dialog*. In XIML, for example, a `dialog element` is defined as being "like a command that can be executed [...] It is the more concrete instantiation of a task." [15]. In contrast, XUL defines a `dialog` as an "element [which] should be used in place of the window element for dialog boxes" [10]. Such ambiguities can easily lead to misinterpretations, especially if users are trained on a particular language and switch to another one.

Mapping a user interface description language to a formal ontology capturing the *semantics* of those terms can avoid such misinterpretations. With the example term *dialog*, a formal ontology can help resolving the ambiguity by indicating that the languages imply different top-level categories such as PROCESS, PLAN, or SOFTWARE COMPONENT as super-category for DIALOG.

Facilitating Extensibility of User Interface Description Languages
XML based languages usually use a fixed set of tags. In order not to be too strictly limited for practical use, many of those languages provide some extension mechanisms such as universal general purpose tags that can be used for user-defined concepts (e.g. the `ELEMENT` tag in XIML). These extension slots are then filled with arbitrary strings.

Arbitrary strings, however, are dangerous. They lead to extensions that are incompatible with each other, interpreted differently by different people and systems
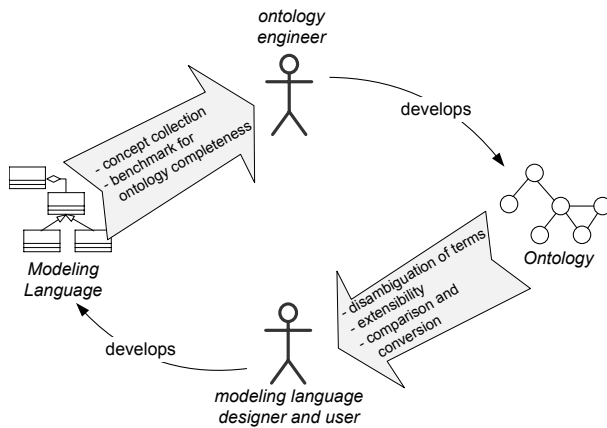
Figure 2. How user interface description languages and ontologies can benefit from each other

relying on different conventions and external documentations, and, in the end, foil the overall idea of having a standardized modeling language.

A formal ontology can help here by providing a standardized vocabulary which can be used to fill such extension slots. Thus, it can be assured that there is an unambiguous interpretation of the extensions.

Model Comparison and Conversion

When bringing together different development teams, information systems, or organizations, it is likely that models created with different user interface description languages already exist. Using a mediating ontology for annotating the models is a common way of establishing comparability between models, not only user interface models [4].

Once models are annotated and can be compared using a common ontology, automatic conversion of models can be long-term objective. For the moment, a common ontology can at least support developers in understanding each other's models and assist them in unambiguously transferring their contents between modeling languages manually.

Fig. 2 summarizes how modeling languages and a formal ontology can benefit from each other.

TOWARDS A FORMAL ONTOLOGY OF THE DOMAIN OF USER INTERFACES AND INTERACTIONS

With these considerations in mind, we have started to develop a formal ontology of the domain of user interfaces and interactions. The goal is to end up with an ontology that is *comprehensive* at least with respect to the expressivity of current user interface definition languages, that is universal enough to be *extendable* to future user interfaces that do not exist at the moment. Furthermore, to support valuable reasoning on user interfaces and provide meaningful semantics, the ontology should be *highly axiomatized.*

To end up with a comprehensive ontology, we have analyzed several user interface description languages in order to collect a maximum set of relevant terms. We have used UsiXML, XIML, UIML, Maria, XUL, LZX, WAI ARIA, and XForms as a basis for identifying the core concepts.

In order to build upon well-acknowledged roots, we have chosen the top level ontology *DOLCE* [9] and its extensions as a basis for our ontology. This top level ontology provides an embracing basic classification of things and has been used as a basis for building numerous ontologies. Since the top level provides a complete classification, it ensures extensibility of the ontology by design, as every new concept can be classified in some existing category. Furthermore, we have reused two core ontologies of software and software components [11], which are also built upon the foundations of DOLCE.

The ontology we have developed is divided into two parts: a *top level* which captures the semantics of the basic terms of the domain, such as *User Interface Component* and *Interaction*, while the *detail level* classifies the actual things that exist in the domain, such as types of user interface components and user tasks that can be performed with those components. The OWL version of the top level ontology consists of 15 classes, two object properties, and 75 axioms, while the detail level consists of 179 classes, eleven object properties, and 448 axioms.

CONCLUDING REMARKS

This position paper has discussed the differences between UI description languages and a formal ontology of the domain of user interfaces and interactions. Furthermore, We have given insight into the development of a comprehensive formal ontology of the user interfaces and interactions domain. In the long run, we are confident that formal ontologies and UI definition languages will both have their places, and that both will benefit from each other.

We have presented a number of potential improvements where developers employing user interface description languages could benefit from those languages being mapped to a formal ontology of user interfaces and interactions. Thus, our claim is that organizations providing user interface description languages could improve the usability and acceptance of those languages by providing such a mapping.

As a long-term objective, such a mapping could even facilitate automatic conversion between models developed with different user interface description languages. To that end, more sophisticated mapping approaches than simply relating elements form a modeling language to a category in an ontology are needed [13].

A formal ontology will not replace user interface description languages, but be a valuable enhancement. Due to the conceptual differences between software mod-

Figure 3. The top level of the ontology of the user interfaces and interactions domain. In the upper part, the design time concepts are shown, the lower part contains the run time concepts. The left part deals with interactions, the right part with components. The white ellipses denote concepts from the reused ontologies (with the following namespace conventions: DOLCE (dolce), Descriptions and Situations (dns), Plans (plans), Functional Participation (fp), Temporal relations (tr), Core Ontology of Software (cos), Core Ontology of Software Components (cosc)), the grey ellipses denote concepts from the top level ontology of the user interfaces and interactions domain. The grey triangles denote definitions carried out in the detail ontology.

els and ontologies, user interface description languages do a better job, e.g., when developing user interfaces in model based approaches. Although there have been attempts for UI code generation from ontologies [8, 17], the latter even claiming that ontologies should entirely replace existing user interface description languages, we believe that a co-existence of both is more beneficial.

Acknowledgements

REFERENCES

1. U. Aßmann, S. Zschaler, and G. Wagner. Ontologies, Meta-models, and the Model-Driven Paradigm. In Calero et al. [3], chapter 9, pages 249–273.

2. C. Atkinson, M. Gutheil, and K. Kiko. On the Relationship of Ontologies and Models. In S. Brockmans, J. Jung, and Y. Sure, editors, *WoMM*, volume 96 of *LNI*, pages 47–60. GI, 2006.

3. C. Calero, F. Ruiz, and M. Piattini, editors. *Ontologies for Software Engineering and Software Technology*. Springer, 2006.

4. J. Fengel and M. Rebstock. Linking Heterogeneous Conceptual Models through a Unifying Modeling Concepts Ontology. In N. Stojanovic and B. Norton, editors, *Proceedings of the 5th International Workshop on Semantic Business Process Management (SBPM 2010)*, volume 682 of *CEUR-WS*, pages 1–4, 2010.

5. M. Fernández, A. Gómez-Pérez, and N. Juristo. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, 1997.

6. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, Juni 1993.

7. J. Guerrero-Garcia, J. M. Gonzalez-Calleros, J. Vanderdonckt, and J. Munoz-Arteaga. A Theoretical Survey of User Interface Description Languages: Preliminary Results. In *LA-WEB '09: Proceedings of the 2009 Latin American Web Congress (la-web 2009)*, pages 36–43, Washington, DC, USA, 2009. IEEE Computer Society.

8. B. Liu, H. Chen, and W. He. Deriving User Interface from Ontologies: A Model-Based Approach. In *ICTAI '05: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, pages 254–259, Washington, DC, USA, 2005. IEEE Computer Society.

9. C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. WonderWeb Deliverable D18 – Ontology Library (final), 2003. `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf`. Accessed August 2nd, 2010.

10. Mozilla. XUL. `https://developer.mozilla.org/en/XUL`, 2010. Accessed August 4th, 2010.

11. D. Oberle, S. Grimm, and S. Staab. An Ontology for Software. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, chapter 18, pages 383–402. Springer, 2nd edition edition, 2009.

12. F. Paternò, C. Santoro, and L. D. Spano. XML Languages for User Interface Models - Deliverable D2.1 of the ServFace Project. `http://141.76.40.158/Servface/index.php?option=com_docman&task=doc_download&gid=5&Itemid=61`, August 2008. Accessed August 9th, 2010.

13. H. Paulheim, R. Plendl, F. Probst, and D. Oberle. Mapping Pragmatic Class Models to Reference Ontologies. In *2nd International Workshop on Data Engineering meets the Semantic Web (DESWeb)*, 2011. to appear.

14. H. Paulheim and F. Probst. Ontology-Enhanced User Interfaces: A Survey. *International Journal on Semantic Web and Information Systems*, 6(2):36–59, 2010.

15. RedWhale Software. The XIML Specification. Available as part of the XIML Starter Kit version 1, available at `http://www.ximl.org/download/step1.asp`, 2000. Accessed August 3rd, 2010.

16. F. Ruiz and J. R. Hilera. Using Ontologies in Software Engineering and Technology. In Calero et al. [3], chapter 2, pages 49–102.

17. K. A. Sergevich and G. V. Viktorovna. From an Ontology-Oriented Approach Conception to User Interface Development. *International Journal "Information Theories and Applications"*, 10(1):89–98, 2003.

18. P. Spyns, R. Meersmanand, and M. Jarrar. Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17, 2002.

19. M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 11:93–136, 1996.

Towards the user confidence in sensor-rich interactive application environment

Ilkka Niskanen
Software Architectures and Platforms
VTT - Technical Research Center of Finland
Oulu, Finland
Ilkka.Niskanen@vtt.fi

Julia Kantorovitch
Software Architectures and Platforms
VTT - Technical Research Center of Finland
Espoo, Finland
Julia.Kantorovitch@vtt.fi

Vildjiounaite Elena
Context-awareness and service interaction
VTT - Technical Research Center of Finland
Oulu, Finland
Elena.Vildjiounaite@vtt.fi

ABSTRACT
The recent advances in sensor-rich, ambient computing environmets have led to a situation in which ordinary users may express negative reactions when they feel that their behaviour is being monitored and analysed by technological systems which they do not understand. Cooking guide is an example application that is heavily depended on dynamic context information and adapts its behavior according to the context data. The VisuMonitor approach, described in this study, supports the users of Cooking Guide by providing visualization views that show the proceeding of cooking processes and also explains the functionality and behavior of the system during different cooking activities, thus improving user awareness, technology acceptance and user education. VisuMonitor utilizes semantic technologies in the modeling of workflows, which facilitates data integration and enables more efficient work progress monitoring and visualization.

ACM Classification Keywords
H.1.2 User/Machine Systems: Human factors.

Author keywords
Context awareness, proactive knowledge, sensors, user education, semantic technologies, user education, data visualization

General Terms
Design, Human factors

1. INTRODUCTION
When evaluating the ideas of sensor-rich, ambient computing environments to ordinary users, non-technical people, in particular, express anxiety when they find themselves in situations, where they feel that their behaviour is being monitored and analysed by technological systems which they do not understand [1]. Such negative reaction to applications which use sensing technology sets a challenge which needs to be addressed. Technology must be regarded as helpful rather than threatening. We believe that if users perceive themselves to understand and to have control over their personal application, they will be more likely to trust applications which use sensing data. Accordingly a knowledge-based system should be able to explain its reasoning, and rules used to justify its conclusions to be accepted by users.

Cooking guide is an example application that is heavily depended on dynamic context information [17]. The Cooking Guide may run in a touch-screen device, for example, and it helps the user during meal preparation by providing detailed, step-by-step explanations. Cooking Guide adapts its behavior according to the context information (e.g. available smart appliances augmented by various sensors, output devices, and user's cooking experience) thus each step can be potentially performed in a different way. Cooking guide is a true effort towards the contextual rich dynamic proactive knowledge-based application. Proactive knowledge base is built from the sensors augmenting the objects in use, surrounding devices and user profiles. Sophisticated data mining algorithms, rule based mechanisms and user model learning techniques facilitate contextual awareness and adaptability towards the assistance and end user ambient support.

The importance of explanation interfaces in providing system transparency and thus increasing user acceptance has been well recognized early in a number of fields such as expert systems [2], intelligent tutoring systems [3], office documents user assistance systems [18], data exploration systems [4], and recommendation systems [5][6][7]. In relation to ubicomp environment, the necessity to support the features that aim at supporting user acceptance by making system's reasoning process visible and insight of the system comprehendible has been acknowledged only recently [1][8][9], while prototyping of such feature is still in its infancy. For our knowledge only work by K.Cheverst [9] has practically addressed the transparency and comprehensibility of the system leveraging the power of explanation user interfaces. There the Intelligent Office System can learn a given user situation to use the inferred rules and support appropriate proactive behaviour such as e.g. turning on/off the fun or opening/closing window under appropriate conditions. On the same time, the system enables the user to explicitly scrutinise and override the 'if-then' rules held in user model. If the user wishes to enquire why the system is performed in a certain way, the appropriate button can be pressed in order to view a window such as the one shown in Figure 1.

Figure 1. Scrutinising the rules behind the prompt me text

However manually acquired textual explanations may not be always sufficient especially in the cases where the context of the application and user is rapidly varying such as in cooking which is a creative process with continuously changing cooking situation, appliances in use and products features. This sets the additional challenges on the design of the user interface. Moreover, the purpose of the system plays an important role in defying of respective elements that influence system acceptance. When interacting with work and task-oriented systems, the perceived usefulness is more important. In contrast when interacting with hedonic systems that are aimed at fun and pleasure (as cooking guide mostly does) the perceived enjoyment is more desirable in achieving user acceptance [10].

2. VISUALIZATION

Looking for the means to fulfil the above discussed requirements, we believe that visualization based aids which are intuitive and easily customizable, may help the user to link the complex contextual world of physical services residing in the environment, reasoning of the system and human mind. Visualization of data makes it possible to obtain insight into these data in an efficient and effective way, thanks to the unique capabilities of the human visual system, which enables us to detect interesting features and patterns in a short time [11]. In particular with recent advances in computer graphics, visualization is able to benefit the sense of wonder connected with the application presenting the content of the data in a completely innovative and quickly comprehendible form.

Currently existing approaches to visualise the rules of the system are targeting mainly application developers [12][19] or data exploitation professionals [13][14][15][16]. Accordingly common for the developed techniques is that they rather support the categorization, browsing and management of potentially complex rule bases, while the ground to the world of physical devices and context attractiveness, fast assimilation and intuitive visualization important for non-technical end user are left beyond.

3. VISUMONITOR – TOWARDS BETTER USER AWARENESS

In this position paper we present a visual monitoring approach – VisuMonitor, which is currently under development. VisuMonitor is directed for the end-users of different context-aware applications and aims towards a better user awareness, technology acceptance and user educating. The approach enhances the sharing of knowledge by integrating information from multiple, heterogeneous sources and providing interactive views to this data. To enable the integration of heterogeneous data sources, VisuMonitor utilizes semantic technologies and especially ontologies that facilitate shared and common understanding of

knowledge domain and are able to describe explicitly the content and semantics of heterogeneous data sources to support integration, processing and further new knowledge discovering tasks. The utilization of semantic technologies provides also an intelligent way to define and use rules that guide the behavior of the application.

The use of semantic technologies is especially pertinent with such applications as the VisuMonitor where complex and heterogeneous data is gathered from multiple sources and it has to be presented to the users in a comprehensive way. The annotation of the data using ontologies and concept taxonomies will allow users to better perceive the relationships between different concepts. Additionally, by utilizing reasoning mechanisms provided by semantic technologies, the data can be better clustered and targeted to the particular users.

VisuMonitor supports the users of Cooking Guide in two ways: showing practical information related to the cooking process itself (the proceeding of the cooking process from one step to another, the information provided by different sensors, the usage of different devices etc.) and providing explanations related to the functionality and behavior of the cooking guide system (for example why the cooking guide application decided to change from speech to textual guidance in some point of the cooking process etc.). VisuMonitor may also educate the user by explaining why the particular recipe/ingredients are recommended e.g. due health reasons, diseases, dietary, recent blood test, etc.

Different cooking processes executed with Cooking Guide are modeled as workflow descriptions. Cooking Guide is tightly integrated with a Workflow engine tool, which manages the workflows that are executed in cooking processes. The executable workflows are described with an XML-based serialization format known as XPDL [20] (XML Process Definition Language). XPDL is a common format supported by a number of editing tools and process execution engines. XPDL workflow models are standardized representations of one or more workflows. The workflow engine plans, checks and manages the execution and states of workflows. If an activity is finished, it is e.g. responsible for checking outgoing conditions of transitions and deciding if the transitions should be activated or not. Workflow engine utilizes also context information extensively. Besides of information source, the engine uses context data to adapt to the situation, to trigger activity transitions and to influence the control flow.

VisuMonitor communicates with Workflow engine to retrieve the necessary information needed for workflow visualizations. In addition to static and dynamic workflow representations, VisuMonitor provides also other workflow related information to the users. It may show, for example, the different resources needed to complete a workflow activity or information related to functionality and behavior of the cooking guide system. By integrating the data acquired from Workflow engine and Cooking Guide, VisuMonitor is able to produce a global view of a cooking process.

3.1 Compositional structure

The compositional structure of the VisuMonitor infrastructure is shown in Figure 2.
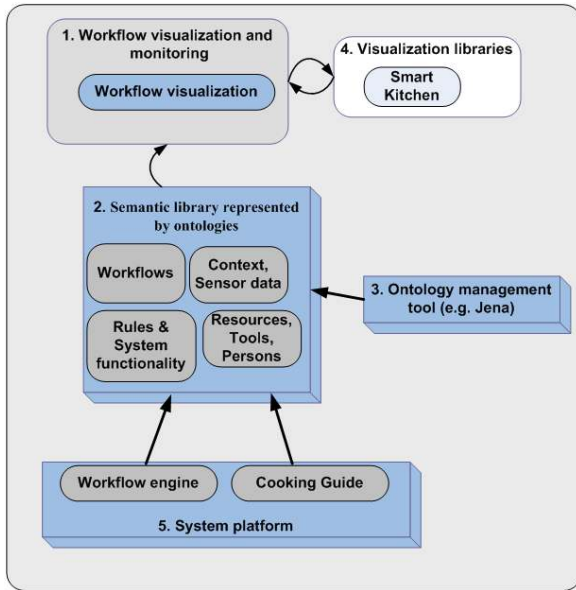
Figure 2. The compositional structure

(1) - Workflow visualization and monitoring, which is a core of the tool. This component provides mechanisms for visualizing workflows and other related information.

(2) - Semantic library represented by ontologies, which will contain the workflow related knowledge base. This component contains semantically modelled workflow descriptions that are visualized with the tool. It may also contain other semantically modelled information, such as context and sensor data, rules and other system functionality data and information about different resources that are related to workflows.

(3) - Ontology management tools, which allow to query and update ontology instances. Some existing open source software like Jena and OWL-API reasoners can be used for this purpose

(4) - Visualization libraries containing domain specific 3D icons that are used in workflow visualizations.

(5) - System platform, which provides the necessary data for workflow visualization. For example, the workflow engine provides static information about workflows and the Cooking guide allows to query such information as the rules applied in the user interface adaptations.

Device/hardware level: from laptop/PC to light device like PDA/smart phone.

3.2 Dynamic structure

While compositional structure provides the static layout of the workflow monitoring architecture, the sequence diagram presented in Figure 3 highlights the way on how different components dynamically interact.
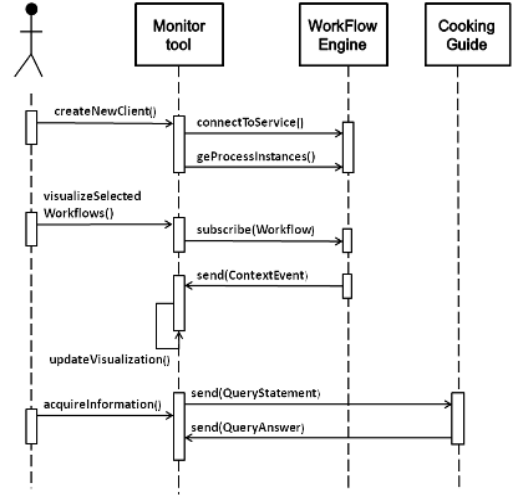

Figure 3. The dynamic workflow visualization

According to the sequence diagram above, the user may first create a client in order to start monitoring workflows. VisuMonitor connects to Workflow engine and retrieves the workflows that are currently hosted by the engine. The user may then select the workflows that he/she wants to visualize and monitor. Subsequently, the monitor communicates with Workflow engine and subscribes as a listener to the selected workflows. As a result, Workflow engine notifies the monitor each time something noticeable happens in the execution of the selected workflows (i.e. a transition from one activity to another or some exception/anomaly occurs during the execution). Each time VisuMonitor receives a change notification it updates the visualization view accordingly. VisuMonitor may also query some additional, workflow related information from the Cooking Guide application. The monitor may acquire, for example, such information as the logical rules applied in a certain cooking activity.

3.3 Semantic data integration

As earlier discussed, VisuMonitor utilizes semantic technologies to provide visually rich and informative workflow representations to the users. For example, by using well defined ontology vocabularies and taxonomic hierarchies data gathered from heterogeneous sources can be better integrated and semantically modeled. For example, when the monitor tool receives non-semantic workflow descriptions, it saves them semantically and annotates the data with descriptive metadata. Next VisuMonitor stores the workflow activities into an RDF data model and finally visualizes the workflows. Whenever additional information is queried from Cooking Guide application, it can be stored into the same RDF model and linked to the appropriate activities of the workflow.

The semantic modeling of workflows has many potential benefits. For example, more comprehensive diagnostics information about the work processes can be produced by discovering the hidden relationships and patterns that may exist in the data. The diagnostics information can include historical, real-time and predictive data. Additionally, the utilization of different reasoning

mechanisms may lead to proactive action recommendations, which in turn enable more efficient fault prevention. Finally, the semantic modeling of data enables more efficient work progress monitoring and visualization. An excerpt from an RDF-description of semantically stored workflow data is presented in Figure 4.

```
<rdf:Description rdf:about="http://owl-ontologies.com/SmartKitchen.owl#ProcessInstanceId3">
  <VisuMonitor:activityDefinitionId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">makeCoffe
  </VisuMonitor:activityDefinitionId>
  <VisuMonitor:activityName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Make Coffee
  </VisuMonitor:activityName>
  <VisuMonitor:priority rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5</VisuMonitor:priority>
  <VisuMonitor:state>CLOSED.COMPLETED</VisuMonitor:state>
```

Figure 4. Example RDF workflow data description

Each of the activities contained by a workflow is defined as an individual, which has certain property and value descriptions. For example, the activity described above has a property 'activityDefinitionId' with value 'makeCoffee' and a property 'state' with value 'CLOSED.COMPLETED'.

3.4 UI design mock-ups

VisuMonitor tool is currently in a design phase and different specifications of the tool are being created. Since visualization and graphical user interface form such an important part of the approach several user interface mock-ups were decided to be created and evaluated before the actual implementation work is started. The purpose of the initial evaluations is to make sure that user perceive the created views and explanation dialogs as informative and comprehensible.

UI design mock-up presented in Figure 5 shows an overall view of the cooking process, in which the proceeding of the workflow from one step to another is illustrated. The already finished activities are depicted with blue boxes, the current step of the cooking process is emphasized with red color and the green boxes represent the activities that have not yet been started. The user is able to acquire more detailed information about different activities by clicking the boxes representing the different steps. The purpose of this kind of overall view is to enhance the general comprehension of cooking processes.
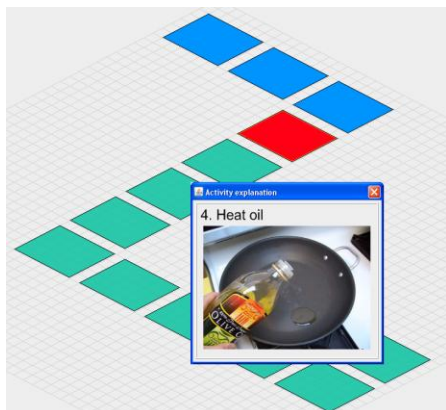

Figure 5. A workflow visualization mock-up

As earlier discussed, a knowledge-based system should be able to explain its reasoning and rules to justify its conclusions.

VisuMonitor addresses this requirement by providing illustrative graphical explanations that makes the behavior of the cooking guide system more transparent. VisuMonitor provides explanations, for example, about the logical rules that guide the functionality of the Cooking Guide system during a certain cooking activity. As an example, a visualization presented in Figure 6 explains one of the rules that automatically turn the Cooking Guide's audio features off if music is detected during the last 20 seconds.

Figure 6. A rule visualization mock-up

Although VisuMonitor is still on a design phase some of the initial user interface mock-ups have been already evaluated in a user study performed for the Cooking Guide prototype [17]. The results proved that VisuMonitor enhances the understanding of application behavior and makes the functionality of Cooking Guide more appreciable for the user.

4. CONCLUSION AND FUTURE WORK

This paper has presented the VisuMonitor approach, which addresses the problem of complex sensor-rich, ambient computing environments causing negative reactions for ordinary users, as they feel they do not have control over their personal applications. VisuMonitor enhances the understanding of application behavior by applying interactive visualization techniques that enable users to observe, manipulate, search, navigate, explore, discover and filter data far more rapidly and far more effectively.

VisuMonitor is tightly coupled with the Cooking Guide application, which provides step-by-step explanations for meal preparation and adapts its behavior according to the context information. VisuMonitor supports the users of Cooking Guide by providing visualization views that show the proceeding of the cooking process from one step to another and also explains the functionality and behavior of the system during different cooking activities. By utilizing different visualization methodologies it aims at improving user awareness, technology acceptance and user education.

An important feature of chosen visualization approach is that it semantically integrates heterogeneous data gathered from different sources. In this way all the workflow related data can be modeled and stored in a similar and structured way. The semantic representation of data facilitates also the discovering of hidden relationships that may exist in the data.

The development of VisuMonitor is currently in its initial stage. The work will continue by analyzing thoroughly the results gained from the evaluation and applying this data in the implementation phase. The construction process will be iterative by its nature and after each design and implementation cycle the approach will be evaluated with the end-users.

Although VisuMonitor is currently developed in a close cooperation with the Cooking Guide application, we are looking for more generic domain independent way to support application users. Different application domain may set an additional research challenge, for example on the visualization aspects like various visualization types might be used depends on the problem domain and also on application features to be monitored and visualized. Additionally, the workflows describing semantic models will be improved by developing the data integration methods and using more sophisticated reasoning capabilities

5. ACKNOWLEDGMENTS

6. REFERENCES

[1] Feeney K. et al. 2008. Avoiding "Big Brother" Anxiety with Progressive Self-Management of Ubiquitous Computing Services, MobiQuitous 2008, July 21-25, 2008, Dublin, Ireland

[2] Klein, D.A. and Shortliffe, E.H. 1994. A framework for explaining decision-theoretic advice, Artificial Intelligence 67, 1994, 201-243.

[3] Sørmo, F. and Aamodt, A. 2002. Knowledge communication and CBR. In Proceedings of the ECCBR-02 Workshop on Case-Based Reasoning for Education and Training, 2002, 487-59.

[4] Carenini, G. and Moore, J. 1998. Multimedia explanations in IDEA decision support system. Working Notes of the AAAI Spring Symposium on Interactive and Mixed-Initiative Decision Theoretic Systems.

[5] Chen, L. and Pu, P. 2005. Trust building in recommender agents. In Proceedings of the Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces at the 2nd International Conference on E-Business and Telecommunication Networks (ICETE'02).

[6] McSherry, D. 2004. Explanation in recommender systems. In Workshop Proceedings of the 7th European Conference on Case-Based Reasoning, 2004, 125-134.

[7] O'Donovan, J. and Smyth, B. 2005. Trust in recommender systems. In Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI'05), 167 - 174.

[8] Callaghan, V., Clarke, G. S., and Chin, S. J. Y. 2008. Some socio-technical aspects of intelligent buildings and pervasive computing research, Intell. Build. Int'l J. 1:1.

[9] Cheverst K., et al. 2005. Exploring Issues of User Model Transparency and Proactive Behaviour in an Office Environment Control System, User Modeling and User-Adapted Interaction 15:235-273

[10] Cramer,H.S.M., Evers V., Van Someren, M., Ramlal, S., Rutledge, L., Stash, N., Aroyo, L., Wielinga, B. 2008. The effects of transparency on perceived and actual competence of a content-based recommender, Semantic Web User Interaction Workshop, CHI 2008, April 2008

[11] Wijk, J. 2005. The value of visualization. Proceedings of the IEEE Visualization (VIS'05), Minneapolis, MN, USA, 23.28 October 2005.

[12] Hassanpour, S., O'Connor, M.J. and Das, A.K. 2010. A Software Tool for Visualizing, Managing and Eliciting SWRL Rules, ESWC 2010, Part II, LNCS 6089, pp. 381–385

[13] Blanchard J., et al. 2007. A 2D-3D visualization support for human-centered rule-mining, Computer and Graphics 31, 3 (2007) 350-360

[14] Ma, Y., Liu B. and Wong, C.K. 2000. Web for data mining: organizing and interpreting the discovered rules using the Web. SIGKDD Explorations, ACM Press, vol. 2, num. 1, pp 16{23}

[15] Hofmann, H. and Wilhelm, A. 2001. Visual comparison of association rules. Computational Statistics, Physica-Verlag, vol. 16, num. 3, pp 399{415}

[16] Lehn, R. 2000. An interactive rule visualization system for knowledge discovery in databases. PhD thesis, University of Nantes

[17] Vildjiounaite E., et al. 2011. Designing Socially Acceptable Multimodal Interaction in Cooking Assistants. *In* proceedings of International Conference on Intelligent User Interfaces, Palo Alto, California, USA, February 2011.

[18] Kohlhase, A., & Kohlhase, M. 2009. Semantic Transparency in User Assistance Systems. In Proceedings of the 27th annual ACM international conference on Design of Communication.Special Interest Group on Design of Communication (SIGDOC-09), Bloomingtion,, IN, United States. ACM Press.

[19] Gribova, V. 2007. Automatic generation of context sensitive help using a user interface project. In proceedings of 8th[h] International Conference "Knowledge-Dialogue-Solutions" – KDS 2007, July 2007, Varna, Bulgaria

[20] WFMC 2002. Workflow Management CoalitionWorkflow Standard: Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMC-TC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA.

Controlling Smart Environments using a Brain Computer Interface

Gernot Ruscher*   Frank Krüger   Sebastian Bader   Thomas Kirste

Universität Rostock,
Albert-Einstein-Str. 21,
18059 Rostock, Germany

*corresponding author: gernot.ruscher@uni-rostock.de

ABSTRACT

We describe first experiments for controlling smart environments using a brain-computer interface. The graphical user interface is automatically synthesised from device models that specify effects of device functions on the environment. Thus, the number of interactions can be reduced, and a novel way of human machine interaction is introduced: Controlling the environment instead of single devices.

Categories and Subject Descriptors

H.5 [**User Interfaces**]: Graphical User Interfaces, Input Devices and Strategies

General Terms

Design

Keywords

BCI, smart environments, graphical user interfaces

1. INTRODUCTION & MOTIVATION

Brain Computer Interfaces (BCI) are ongoing research since the 1970s [10], employing invasive technologies as well as non-invasive approaches such as EEG. Key potential of BCI is the possibility of man-machine interaction without requiring motor activities: Hands free, no gestures, no speech, no pointing and clicking. Recently, low-cost devices have become available at market targeting the gaming scene, claiming, at a very competitive price, to provide the capability of cerebral control for at least a limited set of interactions.

Our experience so far shows that with those simple BCI devices, interaction is kept within tight bounds due to the limits of this communication channel: A merely small character set is available at a low frequency, which leads to a severely limited data rate. Applications based on low-cost BCIs thus have to deal with these limitations and to adapt their graphical user interfaces (GUIs). These need to optimise the number of user interactions necessary to trigger an intended application function. Thus, highly application-specific GUIs need to be implemented, and various approaches have been developed that are aimed at grouping functions *smartly*.

With the vision of Ubiquitous Computing coming true, device become more and more *invisible* to the user, and hence cause the need for novel user interfaces. One specific application field in this context is that of *smart environments* [7]. These build complex sets of heterogenous devices, partly fixed to the environment and partly brought-in by the user. Thus, applications in smart environments need to base on such a dynamic ensemble of devices which are possibly unknown in advance. Developing user interfaces which provide control options for lots of devices with lots of different functions would be a tough task by itself. In addition with the dynamics of the underlying device ensemble it quickly seems to be insolvable.

One approach to provide a user interface for a dynamic device ensemble would be the synthesis of a dynamic GUI from formal device descriptions. Various approaches, for example built upon UPnP [4] or Jini [1], make it possible to generate GUIs, even for the control of a dynamic device ensemble. Unfortunately, users' experience shows difficulties with these approaches.

Our approach presented in this work relies on the following working hypothesis: What users of a smart environment are interested in is not the individual device, but their effect on the environment. One simple example: When a user switches a lamp on, he actually just wants to increase the lightness of the room. In this way, all the lamps of this room are able to increase the lightness and would therefore be redundant with respect to their effects on the environment.

With this article, we describe first experiments to control a smart environment using the neural impulse actuator (NIA), a low-cost brain non-invasive computer interface. We use semantic models of the environment and the devices. We model the devices with respect to their specific influence onto the environment. We present a principle approach for the synthesis of graphical user interfaces in order to reduce the number of necessary interactions.

2. PRELIMINARIES

After presenting the neural impulse actuator, we briefly discuss our lab used for the experiments. Furthermore, we give a short introduction to STRIPS: A formalism to describe preconditions and effects of operations.

2.1 The Neural Impulse Actuator

In 2007, OCZ TECHNOLOGY GROUP INC. [5] introduced the *Neural Impulse Actuator* (NIA): A simple BCI controller, basically a headband, equipped with three electrodes capturing electrical potentials from the forehead. Those potentials include electromyogram (potentials arising from muscle control), electroencephalogram (signals from the nerves in the brain) and electrooculogram (signals coming up during eye movement). A special controller is used to connect the sensors to the computer. The NIA registers itself as a USB human interface device, which basically permits it to act like any other input device, e.g. a keyboard or mouse. Figure 1 shows a photo of the NIA controller.

After calibrating the NIA, it is supposed to be usable as a virtual joystick and to switch events, which can be triggered by different electric potentials or muscle movements. In our experiments, we found the following actions easy and stable to recognise:

- eye movement in general,
- heavy muscle movement on the forehead, or moving the jaw,
- light muscle movement on the forehead,
- heavy thinking, and
- relaxing, or closing the eyes.

Here, we want to use the NIA to control our lab environment, even while working on other subjects. Hence, inputs triggered by heavy thinking and relaxing are not suitable signals for a smart environment controller. However, parallel performance to compose more complex signals does not seem to be helpful, as we want to provide an easy-to-use interface. Therefore, we have at most three distinguishable signals at hand: (i) eye movement, together with (ii) heavy and (iii) light forehead muscle movement. To complicate things further, users of the NIA can perform those signals at a merely low frequency of about 10 per minute at most, leading to a comparatively low data rate.

2.2 Our SmartLab

For our experiments we utilised our SMARTLAB: An instrumented meeting room (cf. fig. 2) equipped with a number of remotely controllable devices. It is frequently used as a

Figure 1: The NIA

Figure 2: Our SmartLab.

room for lectures, presentations, and meetings, but also as an experimental setup for user studies.

Our lab is equipped with a couple of sensors, needed to observe state changes in the room. There are e.g. sensors capable of detecting whether the windows are closed or opened, measuring the current temperature, or detecting persons that enter or leave the room, and estimating their number and current positions [3].

On the other hand there is a number of remotely controllable devices required in typical meeting rooms: Dimmable lamps as well as movable projection screens and sun shades, controllable via EIB [2], a computer video and audio matrix switcher to connect brought-in devices with the installed projectors and audio equipment, just to mention the most important. Those devices are actuators in essence, but can also be seen as specific sensors, in each case providing access to their respective status.

Our lab features a powerful middleware (as for instance described in [6]) which on the one hand allows for control of all existing hardware using simple commands. On the other, besides triggering device actions, our middleware enables every device to make its specific properties accessible to other components in the system.

2.3 STRIPS

To describe the capabilities of devices and their possible actions, we suggest to use STRIPS-operators as illustrated in [9]. Those operators formalise (i) *preconditions* that need to hold to make the execution of the respective operation possible and (ii) *effects* that specify the world state changes provoked by the execution of the respective operation. STRIPS-operators have successfully been used in the context of smart environments before [8].

Due to their associated declarative semantics, they are well suited for an automatic interpretation and hence for the construction of a controller. We annotate every operator with the middleware command which needs to be executed to perform the operation. Figure 3 shows two simple operators describing how to switch a lamp l on and off.

3. OUR APPROACH

As depicted above our user interface needs to cope with a dynamic ensemble of heterogenous devices, which is the reason we do not have the option to hard-code a certain controller for a fixed environment. Therefore, we need to consider ways and means of synthesising a controller from an abstract description of the environment and the devices.

According to figure 4, we consider three different modelling tiers: Top left in the sketch is the layer of the *background model*. It specifies existing parameters of the environment and how they can be modified. To provide an example we have formalised two specific parameters and their respective operations in figure 5: There exists some parameter that corresponds to the *lightness* of our room, and it can be modified by two operations named *increase* and *decrease*. The parameter *temperature* is handled likewise. This explicit kind of formal specification of environmental parameters is needed later on to describe the effects on the environment caused by triggering device actions.

As illustrated in section 2.2 all of our devices are made accessible through our middleware, that way providing on the one hand the entire set of current properties to other components and on the other an easy-to-use interface for triggering device actions. In figure 4 this device representation layer is called *device model* and establishes a certain level of abstraction from the plain hardware, where every device can exist without requiring local knowledge on the existence of other devices, the middleware or even the environment. Besides properties and actions this layer holds additional information on the device such as the device type, its name, or whether the device is currently available in the system.

The third modelling tier, the *effect model*, now establishes the relation between the raw device descriptions from the device model and the environmental parameters from the background model. This is done by modelling device actions with respect to their effect on the environment. Every action is annotated with a formal description of the influence of its execution on the specified environmental parameters. As depicted in figure 3 for instance the execution of the `turnOn` method of a lamp has an effect on the environmental parameter lightness in the form that the latter would be increased. We suggest to use STRIPS as modelling formalism to describe the semantic meaning of actions to the environment.

Figure 4: Integration outline of the proposed controller within the smart environment system.

The information aggregation mechanisms of our middleware enable applications to gather these effect models analogously to the previously mentioned properties as well as type and status information of devices. Therefore a GUI application – called *controller* in figure 4 – is now able to provide itself with a list of all devices together with their descriptions, i.e. the type (lamp, sun shades, ...) of the device and all its available actions, including their particular preconditions and effects. After collecting the operators, it can generate diverse GUIs views. Below we present some first experimental views that shall demonstrate the descriptive power of our proposed approach.

As mentioned above, there are basically three different actions (keystrokes) a human can reliably perform. Based on the formal description of our devices we implemented lab controllers tailored for a limited communication between human and computer to evaluate our approach. We designed them following the Mac Finder's *Column View*. Two keys are used to move the focus up and down a list, the third to select the item.

Our very first prototype contained three columns, of which the first contained a list of device types, the second all available devices of the type selected in the first column, and the third all the applicable functions of the device in the second column. This prototype does not involve any environmental knowledge yet. Due to its three-button-based design, this controller interface would enable users who have to rely on a NIA to take control of a complex and dynamic set of heterogeneous devices once these have been seen by the system through the middleware. But without further tweaking and tuning – which we elaborate on in section 4 – the menus would be very large if they contain every possible action for every possible device.

ACTION(switchOn(l,r))
    PRECOND: Lamp(l) ∧ Room(r) ∧ In(r,l) ∧ Off(l)
    EFFECT: ¬Off(l) ∧ On(l) ∧ Lightness.increase(r)
    COMMAND: `l.turnOn()`
ACTION(switchOff(l,r))
    PRECOND: Lamp(l) ∧ Room(r) ∧ In(r,l) ∧ On(l)
    EFFECT: ¬On(l) ∧ Off(l) ∧ Lightness.decrease(r)
    COMMAND: `l.turnOff()`

Figure 3: An annotated STRIPS-operator describing the switchOn and switchOff actions for a lamp. Every operator contains preconditions and effects of the action, and the command to be executed to perform the action.
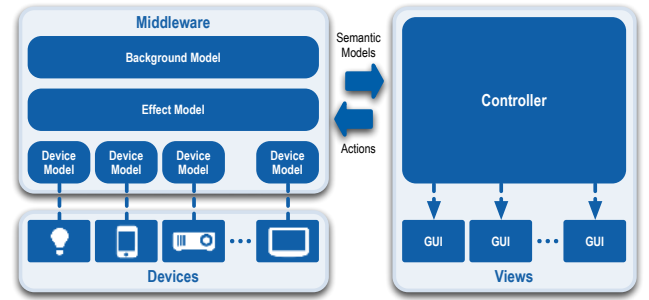

PARAMETER(Room, Lightness)
    OPERATION(Lightness, increase)
    OPERATION(Lightness, decrease)
PARAMETER(Room, Temperature)
    OPERATION(Temperature, increase)
    OPERATION(Temperature, decrease)

Figure 5: Background knowledge: Two environmental parameters and their respective operations.
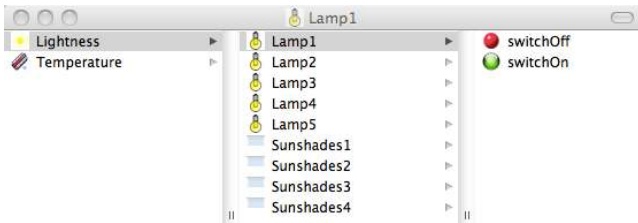
Figure 6: A user interface synthesised by our second controller prototype.

Our second prototype now extracts advantages from our formalisation: As an alternative to a static arrangement based on the device type, we can group our devices with respect to their effects on the environment, as we did in our second controller, which is shown in figure 6. Here, the first column has been replaced by a column containing controllable parameters as for example the lightness, or the temperature. Then, all those devices are listed in the second column which can actually influence the selected parameter. Finally, the third column would again contain performable actions.

But our formalisation of effects has further advantages: The previously depicted controller still displays all the devices even if they have similar influence on the environment. This leads to a long list of devices with each of them still providing every possible action. But, with respect to their effects they are kind of redundant and if we assume that a user is not interested in the device and its particular action itself but is interested in its effect, we can omit all these information and simply provide control options of an environment. For this purpose, we can simply use our existing model of environmental parameters. Our third controller prototype working this way is depicted in figure 7. In the first column it displays the environmental parameters of the background model, which can then be adjusted by selecting one of the items in the second column.

Within our research project MAike, all devices send their descriptions to a central look-up, realised as a tuple space [6]. Furthermore, the NIA controller integrates itself as a new modality for user interaction among other existing ones (speech interaction, intention analysis, ...). So far, we integrated four different device types, with at most eight devices and five actions, and initial experiments showed that those devices are easily controllable using this simple controller together with the NIA.
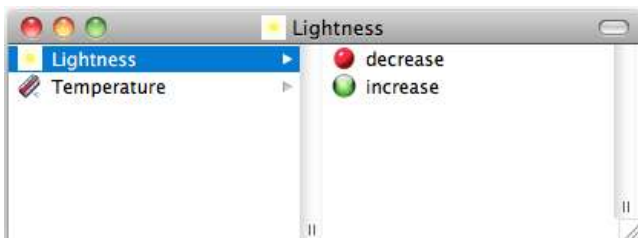

Figure 7: Another user interface synthesised by our third controller prototype.

4. CONCLUSIONS AND DISCUSSION

Our investigations leave us with mixed feelings. On the one hand, it is indeed possible to evoke a limited set of actions using the NIA controller. On the other hand, the concentration required from the user (and, indeed, the level of self control regarding the facial expression), in our opinion leaves ample room for optimisation of both signal acquisition at the sensory level and signal processing at the algorithmic level.

While developing the controller, we could reliably distinguish three different inputs only. Of course the NIA itself provides a richer interface in form of analogue joysticks, but we found that those are hard to control while concentrating on other tasks and they cannot be distinguished as reliable as needed to control the environment. Nonetheless, a better recognition of crisp events would be desirable for the future.

With this work we present a mechanism that deals with the issue of operating a complex and dynamic system through such restricted channels. If we had to deal with an environment that would be static and consist of a fixed number of well-known devices, we would be able to build a controller being perfect in terms of the number of interactions. Because we need an automatically synthesised controller, our approach uses STRIPS operators to apply semantic knowledge of the system's devices and their possible actions aiming at design (and synthesis) of adequate user interfaces.

In the simple approaches presented above, the menus would still grow very large with every new device entering the scene if the menus still contain every possible action for every possible device. Therefore, it is desirable to show the user not the whole menu. Instead, we should offer more abstract actions. Furthermore, we would like those abstract actions to be context dependent and automatically generated.

The focus of this work was not to develop the perfect-working graphical user interface for controlling tasks using the NIA controller. Our goal was to investigate several approaches to reduce the number of interactions a user has to perform to have a function executed. Therefore, the synthesised GUIs will never be the best ones one would find through manual design. This work considers ways and means of involving formal descriptions of environmental knowledge into automatic GUI development. We have not yet evaluated which are best suited for the present application case and different approaches from the ones depicted in this work are imaginable.

As mentioned above, the menu structure used in our prototype, does probably not allow to control larger environments. In the following section 5, we have discussed a number of methods to create more suitable menus and higher level actions. Those have to be implemented and tested with respect to their usability.

Nevertheless, our solution seems to be a principle approach for the synthesis of graphical user interfaces in general, which could bring a significant benefit not only for motorically challenged users, or those who require hands-free interaction in situations, where speech control is not an option.

5. A ROADMAP FOR THE FUTURE

The controller described above is applicable for small well-defined scenarios, but certainly not to control a complex infrastructure with hundreds of device-actions. This is due to the realised menu-based interaction. Considering a usual living room, it is not hard to think of many different devices which should be controllable. Just think of all lamps, shades, multi-media devices, etc. Therefore, alternatives are currently under investigation. We try to learn user preferences online and group devices dynamically. For example, we could group devices together as one *virtual* device if they have been used in one go a couple of times. Another grouping approach could be a Huffman-like encoding which would put the more important devices above others in the list.

As mentioned above lots of user evaluation needs to be done in order to find more intuitive user interface design methods that make use of formal descriptions of the effects caused by device actions on the environment and of the environment itself. One approach could be another control metaphor which is especially designed for brain computer interfaces: A rectangle that periodically rolls over a list of items on the screen and each time marks the currently covered item. The user now simply *thinks* of the particular item he wants to choose and a signal from the brain indicates the moment when the bar covers this item. This selection mechanism can not only be utilised with lists of items, but also like shown in figure 8: First a horizontal bar goes the up-down direction and stops when the height of the selected item is reached. Second, the vertical bar starts moving and is used to indicate the particular device in this line.

The area of automatic *intention analysis*, that is the detection of the user's goals based on his current activities, opens further possibilities. Given the user's current goals, we could automatically re-arrange the menus to provide faster access to device which are most likely to be used in the current situation. Going one step further, high-level actions could also be added to the controller. A room detecting the start of a lecture could offer the compound action like "put my presentation on the projector", which consists of several smaller actions. For this purpose some additional *strategy synthesis* component would be needed.

We have not yet fully covered the possibility of the existence of multiple environments, containing different devices, but controllable through the same user interface. This could be the case in a smart home consisting of several smart rooms, where users possibly would like to have functions performed in other rooms remotely or trigger actions in different rooms simultanously. One straightforward approach would be an additional column inserted before the first one, that specifies the particular room. After this column would be the ones depicted in the previous sections.

So far we have been relying on the feedback on an existing computer screen. This is possible for users that can carry a small display with them. E.g., for people depending on a wheel chair, this monitor can be integrated into it. Alternatives should be investigated, e.g. sounds or small displays right next to the devices which are controllable. This would enable the user to move freely around in the environment without watching a computer screen for every action.
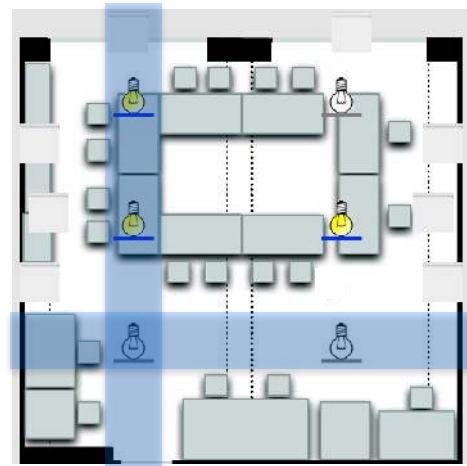

Figure 8: The user interface of a potential controller.

6. REFERENCES

[1] http://www.jini.org, OCT 2009.

[2] http://www.knx.org/, OCT 2009.

[3] http://www.ubisense.de, JUN 2009.

[4] http://www.upnp.org/, DEC 2010.

[5] OCZ Technology Group, Inc. Retrieved from http://www.ocztechnology.com, November 2010.

[6] S. Bader, G. Ruscher, and T. Kirste. A middleware for rapid prototyping smart environments. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, pages 355–356, Copenhagen, Denmark, SEP 2010. ACM.

[7] D. Cook and S. Das. *Smart Environments*. Wiley, 2005.

[8] C. Reisse and T. Kirste. A distributed action selection mechanism for device cooperation in smart environments. In *Proceedings of the 4th International Conference on Intelligent Environments*, Seattle, USA, 2008.

[9] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition edition, 2003.

[10] J. Vidal. Toward direct brain–computer communication. In *Annual Review of Biophysics and Bioengineering*, pages 157–180, 2 1972.

Automated Ontology Evolution as a Basis for Adaptive Interactive Systems

Elmar P. Wach

STI Innsbruck, University of Innsbruck/
Elmar/P/Wach eCommerce Consulting
Technikerstraße 21a, 6020 Innsbruck,
Austria/ Hummelsbüttler Hauptstraße
43, 22339 Hamburg, Germany
+49 172 713 6928

elmar.wach@sti2.at/
wach@elmarpwach.com

ABSTRACT

The research presented in this paper aims at realising an automated ontology evolution process based on feedback without a human inspection. For that, a generic adaptation strategy consisting of a feedback transformation strategy and an ontology evolution strategy is formulated. It decides when and how to evolve by evaluating the impact of the evolution in the precedent feedback cycle. These strategies are implemented in a feedback transformer component and an adaptation manager component respectively, constituting a new adaptation layer. The adaptive ontology is evaluated with an experiment and validated with a real-world conversational content-based e-commerce recommender system as use case.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory – *graph algorithms, graph labeling*. H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *relevance feedback*. H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *commercial services, web-based services*. I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *representations (procedural and rule-based), semantic networks*. I.2.6 [**Artificial Intelligence**]: Learning – *concept learning, knowledge acquisition*. K.4.3 [**Computers and Society**]: Organizational Impacts – *automation*

General Terms

Algorithms, Management, Measurement, Design, Experimentation, Standardization, Languages.

Keywords

Ontology Evolution, Ontology Versioning, Recommender Systems, Self-Adapting Information Systems, Algorithms.

1. INTRODUCTION

Today, the user in the Internet gets overflowed with information and products that she should purchase. Not only becomes it difficult for her to take the right buying decision, but also don't match many search results her needs. Hence, recommender systems in e-commerce applications have become business relevant in filtering the vast information available in the Internet (and e-shops) to present useful search results and product recommendations to the user.

As the range of products and customer needs and preferences change – and they will change even more frequently – it is necessary to adapt the recommendation process. Doing that manually is inefficient and usually very expensive.

Therefore, this research proposes an automated adaptation of the recommendation process by utilising semantic technology and processing user feedback.

The shortcomings of a manual adaptation of the recommendation process based on user feedback are aimed to be solved with a system based on product domain ontologies (PDO) modelling the products offered in the e-commerce application and automatically evolving with processing user feedback. As the PDO describes the products formally, it offers a higher computability than conventional product descriptions and, hence, facilitates automated processing of information.

In order to get the system user-driven, user feedback is gathered by unobtrusively monitoring user needs. The more information is available from a user, the better the adaptation to her needs can be. Hence, implicit and explicit feedbacks provided via feedback channels are evaluated. Implicit feedback is given by the user as a side-effect of her usage behaviour, e.g. by clicking on the product recommended. Explicit feedback could be provided by answering questions about her satisfaction with the application. As this effort cannot be expected from a user, an alternative is to extract feedback from the Web that could also deliver new information and aspects about the products offered. In order to focus this research on developing an automated ontology evolution, the feedback is assumed to be given.

On a more abstract level, this research aims at realising an automated ontology evolution process based on feedback without a human inspection.

Topics of the SEMAIS 2011 workshop related to this research:

- What are the major technical challenges for developing or generating user interfaces based on semantic models? This paper aims to answer the above question with a generic approach.
- For which kind of systems or applications are semantic models particularly useful?
  The use case in this paper is a recommender; for which other systems or applications can it be useful?
- Additional question: Which ontological information and its changes (properties, etc.) are requested by adaptive interactive systems?

2. RELATED WORK

Previous approaches to the topic of this research can be found in concepts for ontology evolution like formulated frameworks for ontology evolution, e.g. [6], [7], [8], [14], [16], [18]. Due to the specific challenges of the present research like the automated ontology evolution process, none of the identified frameworks can be completely used as basis, e.g. all of the frameworks include a step for the human inspection of the ontology changes before they are executed. The closest work to the research in this paper is [16] – in the six phase evolution process, two steps include manual activities, namely (i) "Implementation" in which the implications of an ontology change are presented to the user and have to be approved by her before execution, and (ii) "Validation" in which performed changes can get manually validated. The research in this paper proposes an extension of [16] towards an automated ontology evolution by developing a generic adaptation strategy and further introducing a complete feedback cycle based on the ontology usage that eliminates the implementation and validation steps of above – an ontology change needs those manual steps no longer, as an insufficient change would be alerted by a negative feedback and get corrected automatically.

The approaches to the identified recommender systems [1], [2], [4], [11], [12], [13] research the impact on the recommendation result by using the different recommender types (i.e. content-based filtering, collaborative filtering, hybrid approaches) and mostly utilising domain and user ontologies, whereas the feedback gets processed in the latter one. None of them combines an e-commerce domain ontology with the processing of implicit and explicit user feedbacks.

3. ADAPTATION STRATEGY

For realising an automated ontology evolution, a generic adaptation strategy consisting of a feedback transformation strategy and an ontology evolution strategy is formulated. It decides when and how to evolve by evaluating the impact of the evolution in the precedent feedback cycle. The first question defines the (temporal and causal) trigger initiating the ontology change. Basically, this is receiving and transforming the feedback into ontology input and will be addressed with a feedback transformation strategy (confer chapter 3.1).

The second question defines the changing of the ontology including instance data. This is denoted by ontology evolution referring to the activity of facilitating the modification of an ontology by preserving its consistency [19]. This will be addressed with an ontology evolution strategy (confer chapter 3.2)

considering also how identified conflicts can be solved, e.g. when moving a sub-concept.

By following the principles of adaptive systems [3], the adaptation strategy is implemented in a new adaptation layer consisting of components in which the user feedback gets transformed (i.e. Feedback Transformer) and the respective actions are decided and initiated (i.e. Adaptation Manager).

3.1 Feedback Transformation Strategy

In order to automatically process feedback, i.e. transforming it into ontology input, an adequate feedback transformation strategy has to be formulated and implemented. It has to allow for different feedback channels as well as different kinds of feedback. This strategy is implemented in the feedback transformer component depicted in figure 1. In the Feedback Transformer the ontology affected by the feedback reported is identified, the feedback is analysed and transformed, and eventually get related to the precedent feedback.
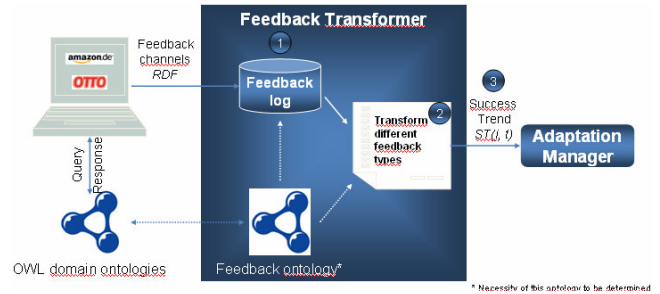

Figure 1. Conceptual architecture of the feedback transformer component

Basically, the strategy comprises the following steps:

1. Gather feedback from the different channels
2. Transform different feedback types
1. Report transformed feedback to the next component

Ad 1. Each feedback channel provides user feedback as RDF triples at separate SPARQL endpoints. The RDF triples are retrieved by the Feedback Transformer and captured in a semantic feedback log as instances of the feedback ontology (confer next paragraph).

Ad 2. The feedback ontology is a prerequisite for the meaningful analysis of the feedback [17]. In the present research, it models the feedback at the product level and additionally contains all product names of the product ontologies. The structure of the feedback ontology enables reasoning about a product and its ratings including the historical development as well as identifying properties and relations to be newly added to the product ontology. Accordingly, we distinguish between the three feedback types "KPI[1] trend", "product rating", and "new property". The root concept is "Feedback". Its hierarchy consists of the sub-concepts "KPI trend", "product rating", and "new property". Appropriate relations like "previousRating" model the history of the ratings.

---

[1] Key Performance Indicator, measured in the application layer

The first two feedback types are converted by either a simple transformation or a feedback evaluation algorithm to values in the range [+1…-1] relating the current transformed feedback to the one in the precedent cycle.

For the feedback type "product rating" the RDF feedback includes the product name and rating but no new potential property. The feedback is transformed with a feedback evaluation algorithm. In the first step, the impact of the ontology evolution on the KPI (e.g. conversion rate and click-out rate) is calculated for each product and feedback channel. In the next step, all feedback channels are aggregated at the product level. Finally, a trend metric is calculated relating the current transformed feedback to the one in the precedent cycle.

For the feedback type "new property" the RDF feedback includes the product name and a new potential property to be eventually added to the product ontology, e.g. information like aspects or relevant features of a product. This feedback type is not covered by the feedback evaluation algorithm. A new sub-property for the aspect/ feature is created in the feedback ontology and its count gets related to the count of all properties in the respective PDO. When reaching a defined threshold, the new property is added to the respective PDO.

The semantic feedback log captures the exact sequence of the reported feedbacks. Each feedback is associated with the respective product (i.e. the RDF feedback contains the corresponding product name) and represented as instances of the sub-concepts of "Feedback". These instances contain the product name, feedback channel, date and time of the feedback, rating, and the certainty of the rating as well as the number of properties contained in the product ontology. The log allows the analysis of the feedback development.

Ad 3. After having transformed the different feedback types, the calculated metrics relating the current feedback to the feedback in the precedent cycle are reported to the next component, i.e. the Adaptation Manager.

3.2 Ontology Evolution Strategy

The ontology evolution strategy defines how the PDO change. It associates the transformed feedback values to evolution actions and ensures a consistent new version of a PDO. This strategy is implemented in the adaptation manager component depicted in figure 2. In the Adaptation Manager the structure of the respective ontology get dynamically analysed with SPARQL SELECT statements and the ontology changes (e.g. switching individuals, switching annotation property labels and comments, changing annotation property priorities, adding new properties) are executed with SPARQL CONSTRUCT rules according to predefined evolution strategies.
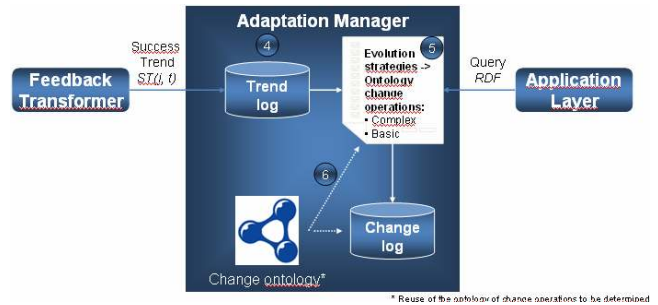

Figure 2. Conceptual architecture of the adaptation manager component

Basically, the strategy comprises the following steps:

4. Gather feedback trends
5. Associate ontology changes with evolution strategies
6. Ensure a consistent ontology evolution

Ad 4. In each feedback cycle the transformed feedback gets reported to the Adaptation Manager. The feedback is based on the product level. Each reported feedback is captured in a trend log at the product level.

Ad 5. The central task of the ontology evolution strategy and the Adaptation Manager is to choose the right evolution, i.e. ontology changes, for the transformed feedback.

[9] introduced a meta-ontology for the ontology evolution enabling representation, analysis, realisation, and sharing of ontological changes. Each possible change is represented as a concept in that evolution ontology having an evolution log as instance capturing the changes. A central element in the framework of [7] are a change log and an ontology of change operations for OWL describing basic ontology change operations[2] and complex change operations composed of multiple basic operations. This research aims at utilising the ontology of change operations sketched above.

Derived from user scenarios, evolution strategies are defined reflecting different behaviours and associating ontology changes, namely:

- Risky Evolution ("always evolve differently"): Regardless of the feedback trend between two consecutive feedback cycles, other complex ontology change operations are executed

- Progressive Evolution ("learn from the past"): Depending on the leap of the trend, same or different complex ontology change operations are executed; in case of a negative trend, it is optional to either do a different complex ontology change operation or a rollback; additionally, with a threshold indicating the increase of the trend between the current and the precedent cycle the "risk" of the evolution can be adjusted and the strategy tuned towards the Risky Evolution (with a higher threshold)

- Safe Evolution ("only revert negative trends"): In case of a negative trend, a rollback is executed

---

[2] Basic ontology change operations modify only one specific feature of an OWL ontology

- Rollback ("undo the ontology changes"): Reverts the ontology changes from the precedent feedback cycle and is based on any reason or decision of the manager; it is executed only once but can be manually chosen multiple times

Ad 6. After having chosen the ontology change operations to be executed, the ontology has to evolve depending on rules and by retaining its consistency to finally provide its knowledge to the application layer.

The existing research about ontology evolution is based on the work about data schema evolution but focuses on the specific needs of ontologies, e.g. [10], [15], [16].

To execute ontology changes, an ontology evolution algorithm has to be formulated. The following prerequisites have to be respected:

- The basic and complex ontology change operations have to be defined formally

- It has to be defined when an ontology is inconsistent, i.e. an ontology consistency model has to be formulated; the preconditions and postconditions of the change operations have to be checked before execution

- The options for a consistent ontology evolution have to be identified and the "best" evolution path chosen; in the present research the belief revision principle of minimal change will be followed [8]; eventually, the ontology evolution algorithm can be formulated

When evolving the ontology, it has to be clear how the ontology has been evolved over time, i.e. the different ontology evolutions have to be versioned. In the context of this research this is of paramount importance, for (i) the ontology changes in the current feedback cycle are derived from the changes in the precedent cycle and (ii) an undoing of the changes in the precedent feedback cycle, i.e. a rollback, has to be realisable.

The preferred concept of ontology versioning is change-based versioning (i.e. each state gets its own version number and additionally stores information about the changes made), because it facilitates change detection, integration, conflict management [9], and it allows the interpretation how ontology changes influence the KPI. A change-based versioning can be best realised by tracking the ontology changes in a semantic log [9].

The change ontology models the applicable changes and meta-information and provides the semantics of all possible ontology changes. The root concept is "Change". Its hierarchy consists of the sub-concepts "complex ontology change operations" and "basic ontology change operations". Appropriate relations like "previousChange" model the history of the ontology changes and construct the sequence of the required changes. The structure of the change ontology enables reasoning about changes including their historical development.

The semantic change log captures the exact sequence of the ontology changes executed. Each change is represented as instances of the sub-concepts of "Change". The log allows the analysis of the change development including realising a rollback.

The whole adaptation strategy and its implementation via the components Feedback Transformer and Adaptation Manager allow eliminating both manual steps in the six phase evolution process of [16]:

- Phase "Implementation" (ontology changes are manually approved before execution): Nobody has to do that, as the ontology evolution is seen as a complete feedback cycle – an insufficient ontology change is indicated by decreased KPI and gets revised according to the evolution strategy chosen

- Phase "Validation" (performed changes can get manually validated): As the ontology changes are predefined, only valid changes are executed, and nobody has to validate them

4. EVALUATION AND VALIDATION

The automatically evolved ontology is going to be compared with a manually evolved one by setting up and evaluating an experiment with ontology experts. Those analyse the feedbacks delivered and decide the ontology changes to be executed. Eventually, the ontology resulted from this manual evolution is compared with the automatically evolved one regarding the evaluation criteria consistency, completeness, conciseness, expandability, and sensitiveness [5].

The validation of this research is done with a use case by utilising a real-world conversational content-based e-commerce recommender system and two feedback channels – the Web application and information extracted from Linked Open Data. As the recommender is already used in live e-commerce applications, the evaluation of the system adaptations is a real-world scenario.

The recommender is based on PDO that semantically describe the products offered in e-commerce applications according to the GoodRelations ontology.[3]

The success of such a system is usually defined by analysing KPI like the achieved conversion rate (i.e. customers-to-recommender users ratio) or click-out rate (i.e. clicks-to-recommendations ratio).

The evaluation scenario is to test and evaluate the impact of the ontology evolution by utilising the formulated evolution strategies, i.e. Risky Evolution, Progressive Evolution, and Safe Evolution.

The impact of the ontology evolution will be analysed and evaluated with regard to the respective KPI at the application level after each to be defined number of accomplished recommendation processes and reported to the ontology.

According to the respective results and feedbacks reported, the ontology evolves. The ontological knowledge is provided to the application layer, and eventually adapted recommendations are presented to the customer. The feedback circle of the automated system concludes with re-evaluating the KPI after having again reached the defined number of recommendation processes.

The intended results are a highly adaptive system and eventually better recommendations given to the user leading to an increase of the defined KPI. The expected business impacts are a higher

---

[3] www.purl.org/goodrelations

customer satisfaction and loyalty and eventually increased revenue for the provider of the application.

This evaluation procedure will be executed for all three evolution strategies and evaluated analogously.

An interesting result of the evaluation scenario would be that one of the three evolution strategies leads to a higher increase of the KPI.

In case a predominant evolution strategy is identified, it can be interpreted that the historic development of changing the ontology (i.e. doing the same change again versus doing a different change) has a significant influence on the customer satisfaction. Though, this can in the case of same changes only be valid within a realisable frame, e.g. it is not possible to move up a sub-concept in the concept hierarchy infinitely times.

5. CONCLUSION
The need for automatically updating and evolving ontologies is urging in today's usage scenarios. The present research tackles an automated process for the first time (to the best knowledge of the author). The reason for that can be found in the ontology definition "formal, explicit specification of a shared conceptualisation". "Shared" means the knowledge contained in an ontology is consensual, i.e. it has been accepted by a group of people. Entailed from that, one can argue that by processing feedback in an ontology and evolving it, it is no longer a shared conceptualisation but an application-specific data model. On the other hand, it is still shared by the group of people who are using the application. It may even be argued that the ontology has been optimised for the usage of that group (in a specific context or application) and, hence, is a new way of interpreting ontologies: They can also be a specifically tailored and usage-based knowledge representation derived from an initial ontology – an ontology view, preserving most of the advantages like the support of automatically processing information. Thus, this changed way of conceiving ontologies could facilitate the adoption and spread of using this powerful representation mechanism in the real world, as it is easier to accomplish consensus within a smaller group of people than a larger one.

6. ACKNOWLEDGMENTS

7. REFERENCES
[1] Aktas, M. S., Pierce M., Fox, G. C., Leake D. 2004. A Web based conversational case-based recommender system for ontology aided metadata discovery, *Proceedings 5th IEEE/ ACM International Workshop on Grid Computing*, pp. 69-75.

[2] Blanco, Y. et al. 2005. AVATAR: An approach based on semantic reasoning to recommend personalized TV programs, *Proceedings 14th International conference on World Wide Web*, pp. 1078-1079.

[3] Broy, M. et al. 2009. Formalizing the notion of adaptive system behavior, *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC '09)*, pp. 1029-1033.

[4] Drachsler, H. et al. 2008. Effects of the ISIS recommender system for navigation support in self-organised learning networks, *Proceedings of Special Track on Technology Support for Self-Organised Learners*, pp. 106-124.

[5] Gómez-Pérez, A. 2001. Evaluation of ontologies, International Journal of Intelligent Systems, Volume 16, pp. 391-409.

[6] Haase, P. et al. 2005. A framework for handling inconsistency in changing ontologies, *Proceedings of the 2005 International Semantic Web Conference (ISWC05)*, pp. 353-367.

[7] Klein, M. and Noy N. F. 2003. A component-based framework for ontology evolution, *Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems*.

[8] Konstantinidis, G. et al. 2007. Ontology evolution: A framework and its application to RDF, *Proceedings of the Joint ODBIS & SWDB Workshop on Semantic Web, Ontologies, Databases*.

[9] Mädche, A. et al. 2002. Managing multiple ontologies and ontology evolution in Ontologging, *Proceedings of the IFIP 17th World Computer Congress – TC12 Stream on Intelligent Information Processing*, pp. 51-63.

[10] Mädche, A. et al. 2003. Managing multiple and distributed Ontologies on the Semantic Web, *The VLDB Journal – The International Journal on Very Large Data Bases*, Volume 12, Issue 4, pp. 286-302.

[11] Maidel, V., Shoval, P., Shapira, B., Taieb-Maimon, M. 2008. Evaluation of an ontology-content based filtering method for a personalized newspaper, *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 91-98.

[12] Middleton, S. E., De Roure, D. C., Shadbolt, N. R. 2001. Capturing knowledge of user preferences: Ontologies in recommender systems, *Proceedings 1st international conference on Knowledge capture*, pp. 100-107.

[13] Middleton, S. E., Shadbolt, N. R., De Roure D. C. 2003. Capturing interest through inference and visualization: Ontological user profiling in recommender systems, *Proceedings 2nd international conference on Knowledge capture*, pp. 62-69.

[14] Noy, N. F. et al. 2006. A framework for ontology evolution in collaborative environments, *Proceedings of the 2005 International Semantic Web Conference (ISWC05)*, pp. 544-558.

[15] Plessers, P. 2006. *An approach to Web-based ontology evolution*, Ph.D. Thesis, Department of Computer Science, Vrije Universiteit Brussel.

[16] Stojanovic, L. et al. 2002. User-driven ontology evolution management, *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW '02)*, pp. 285-300.

[17] Stojanovic, N. and Stojanovic, L. 2002. *Usage-oriented evolution of ontology-based knowledge management systems*, LNCS 2519, pp. 1186-1204.

[18] Stojanovic, N. et al. 2003. *The OntoManager – a system for the usage-based ontology management*, LNCS 2888, pp. 858-875.

[19] Suárez-Figueroa, M. C. and Gómez-Pérez, A. 2008. Towards a glossary of activities in the ontology engineering field, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC '08)*.