Supporting Meetings with a Goal-Driven Service-Oriented Multimedia Environment

Karin Anna Hummel, Wolfgang Jochum, Stefan Leitich, Bernhard Schandl

University of Vienna, Faculty of Computer Science
Department of Distributed and Multimedia Systems
Liebiggasse 4/3-4, A-1010 Vienna, Austria

{karin.hummel,wolfgang.jochum,stefan.leitich,bernhard.schandl}@univie.ac.at

ABSTRACT
During the last decade the number of personal multimedia-enabled devices has increased significantly in everyday usage. Additionally, high-end devices for multimedia environments enable multi-modal human computer interaction and in particular advanced collaboration. Most multimedia environments focus on efficient provisioning of multimedia services but show a lack of user-centric aspects, like explicit reasoning about the users’ wishes and fulfillment of requirements.

In this paper, we present a novel goal-driven approach for multimedia service composition which is able to reason about the users’ demands and to adapt to new context situations in a non-intrusive manner. The fulfillment of goals is assured by hierarchical goal structuring, service provisioning, and evaluation of service fulfillment degrees. We apply this approach to a typical multimedia-enriched meeting scenario described by means of Semantic Web ontologies. A flexible and modular service-oriented software architecture demonstrates the usability of the envisioned companion-like smart meeting room.

Categories and Subject Descriptors
H.1.2 [Information Systems]: Models and Principles—User/Machine Systems; H.5.1 [Information Systems]: Information Interfaces and Presentation (e.g., HCI)—Multimedia Information Systems; I.2.4 [Computing Methodologies]: Artificial Intelligence—Knowledge Representation Formalisms and Methods

General Terms
Design, Human Factors

Keywords
Goal-Driven Architecture, Semantic Web, Service-Oriented Architecture

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MSC’05, November 11, 2005, Singapore.
Copyright 2005 ACM 1-59593-245-3/05/0011 ...$5.00.

1. INTRODUCTION
In the field of user interactive systems, five major trends have emerged during the last few years: (1) increasing pervasiveness of computing power and sensors, (2) the user-orientation implemented by services and configurable devices, (3) the increasing digitalization of business workflows, such as meeting situations, (4) the recognition of semantics and semantically-enriched data and process models as an enabling factor for interoperable systems, and (5) the emergence of self-adapting and learning systems.

Due to the complexity of computing and hardly manageable direct interactions, the approach of goal-driven user interaction will – in our belief – gain more and more importance. To increase system acceptance, users should only specify what they want to do while the complexity of how to operate the service should be hidden to a high degree. In our vision, we substitute the simple interactive selection of application functions by an omnipresent, but non-intrusive and non-disturbing assistant for multimedia environments.

We propose a system design where three types of knowledge are used to support user requirements: (1) general knowledge about life situations, such as a meeting, (2) situation-related knowledge (e.g. the agenda of a particular meeting), and (3) live knowledge, i.e. data gathered by sensors, external events, and user interaction, including the data usually denoted as context. The sources of these knowledge types and some examples are depicted in Figure 1.

Figure 1: Types of knowledge

Based on these knowledge types, we propose a system to integrate services, devices, and users in a flexible and adaptive manner. In order to achieve adaptability of the system, means for modeling the success of operations are required. Therefore, we propose a goal-driven system archi-
tecture which allows to model workflows as goal hierarchies. Services are used to fulfill goals which cannot be decomposed any further. A matching algorithm selects the service capable of fulfilling the particular goal in a given context best. This separation between complex goal hierarchies and simple services exhibits some major advantages. First, the approach allows to model situations where different services are capable of fulfilling a specific goal, which makes the approach robust. Services can easily be added without additional system changes, which makes our approach scalable. Similarly, services do not have to be altered in case new goals are introduced. Finally, since services are evaluated depending on the given context, the system further becomes context-aware without changing its purpose expressed by the goal hierarchy.

For demonstration purpose, we apply the approach to a goal-driven, service-oriented smart multimedia environment. We choose the meeting application domain because it is an ideal prototype and demonstration object for various interesting aspects. For example, in a meeting, persons with different organizational backgrounds collaborate for a foreseeable period of time. The meeting participants often bring in their own personal devices, which are probably unknown to the system. The semantic models of meetings (i.e. data objects, processes, artifacts, etc.) are well-known and can be reliably expressed and modeled. Furthermore, the concept of a goal is inherent in the meeting situation, since economics prohibits to hold meetings that do not produce results, or at least strongly aim to do so. The envisioned smart meeting room will be capable of tracking activities, suggesting appropriate alternatives, and automating user interactions. It will play the role of a meeting companion.

Synopsis

The remainder of this paper is organized as follows: After discussing related work in section 2, Section 3 introduces the proposed architecture of our system and its main system components. In Section 4, we present a formal model to describe goals, their attributes, and their relations. Section 5 describes how services are modeled, while Section 6 describes the process of matching between goals and services. To illustrate our concept, Section 7 presents thoughts for the application of the presented approach in specific meeting situations.

2. RELATED WORK

The ubiquitous enrichment of working and home environments with multimedia-enabled devices has been investigated in various studies. For such environments, the architecture DCOD/VDSG [9] proposes to achieve two main benefits: (1) The devices available in a multimedia environment can be controlled by the use of a single mobile device, like a PDA, and (2) services and functionalities are composed by combining the environment’s devices into virtual devices. In contrast to the user interaction driven control, we propose an approach based on the companion metaphor, which allows to hide system complexity from the user to a high degree. Baudara et al. [2] propose an ontology for the semantic description of devices. This approach addresses the physical level of device description (including hardware and software properties) and does not consider the problem of matching device descriptions to actual user or system requirements. In this work, we describe how the matching between user requirements modeled by goals and services can be supported by using extended ontologies.

The RUBI framework [11] emphasizes the similarity between resource discovery and network routing, since both fulfill the need to discover and locate services and resources available over the network. In contrast to service discovery in a highly dynamic network, we focus on a system architecture supporting service matching to a particular requirement and context. Heider and Kirste [12] present a promising approach to goal-based interaction with complex devices in the home infotainment area. They state the important point that a user is not primarily interested in functions, but in goals. Interaction with devices should be modeled in order to support this assumption. In [7], the authors present a smart meeting room based on a context broker architecture, which emphasizes the context aspect of a meeting scenario. Similar to our approach, the authors use the Semantic Web as source for key technologies. The system architecture presented in this article extends these approaches by selecting services while considering both the goals and the context.

3. ARCHITECTURE OVERVIEW

The success of a smart multimedia meeting room relies on the fulfillment of the meeting participants’ requirements in terms of services provided. Although general meeting requirements may persist, services may change due to technological progress, and new services might be needed over time. Therefore, flexibility and ease of enhancement are major quality criteria of the proposed system. In our architecture, we combine two perspectives. First, we base the architecture on general meeting workflows as described in Section 1. Second, we introduce multimedia meeting services following the approach of Service Oriented Architectures (SOAs) [15]. The services of this smart meeting room are distributed and accessible via different stationary and mobile devices. For the latter, the meeting room should provide wireless access, for example, via WLAN and Bluetooth.

The multimedia environment system consists of four main components as depicted in Figure 3: (1) the agenda preparation module, (2) the goal definition engine, (3) the runtime environment, and (4) the learning & self-adaptation module. Additionally, a repository stores and manages all data related to the system such as, for example, sensor data, items of the meeting agenda, and human computer interactions during the meeting. The object space serves as a virtually shared meeting memory, and various services provide multimedia meeting support.

3.1 Agenda Preparation

The agenda preparation module supports the person who conducts the meeting (typically the moderator) in all activities preceding the meeting. A proactive assistant helps the moderator to create and structure the agenda. We further include tasks that are not directly related to the agenda, like the invitation of meeting participants or the booking of required equipment.

The structured agenda is the source of the definition of goals. These goals can be derived from the agenda items or from combinations of such items. For instance, based on the descriptive agenda item “presentation of annual revenue numbers” we can infer the goal “presentation of annual revenue numbers is successfully finished” and related subgoals,
The agenda is specified using vocabulary of the agenda ontology, which is a representative of the first type of knowledge (general knowledge). The agenda ontology defines a vocabulary that allows to model all relevant aspects of a meeting, including participants, topics, and additional documents. It defines concepts for several well-established meeting agenda items, like presentation, discussion, negotiation, or coffee break. The agenda ontology and corresponding individuals are defined in OWL DL\(^1\) in a flexible, extensible, and exchangeable manner.

### 3.2 Goal Definition

Using the input from the agenda preparation module, the goal definition engine defines a goal hierarchy, which includes additional information provided by the meeting initiator and lessons learned from previous meetings extracted from the repository. The goal hierarchy may directly be modified by the initiator, guided by a proactive assistant agent. Together with the agenda the goal hierarchy is used by the runtime environment as script for the actual meeting. We define a goal ontology, which provides a vocabulary to define goals and their relations. The formalism of goal modeling is described in Section 4.

### 3.3 Runtime Environment

The runtime environment supports the meeting participants during the entire meeting. Its versatile functionalities are divided into several submodules, which communicate by means of an event-based distribution system.

The agenda processing component uses the previously defined agenda, and assists the meeting moderator in managing the meeting. It allows for short term changes of the agenda and associates events with the appropriate agenda items. It keeps track of the time budget for each agenda item and permanently informs about the meeting progress. The goal supervision component acts as counterpart to agenda processing. It uses the meeting’s goal hierarchy and controls the degree of fulfillment for each defined goal. Similar to agenda processing, goals can be changed during the meeting. However, late changes cause heavy-weight management operations due to the restructuring of the goal hierarchy. Furthermore, participants and the moderator might want to overrule automatic goal fulfillment, which is also achieved by human initiated changes during runtime. The goal supervision and agenda processing modules are closely related.

The moderator is further supported by the moderation support component, which manages interactions with the meeting moderator and provides the interface to all functionalities available within the runtime environment. The matching of goals to multimedia services provided by the smart meeting room is performed by the goals/services broker. It infers the meeting’s requirements from the agenda and the goal hierarchy (and includes additional requirements defined by the meeting moderator and/or meeting participants). For each goal the broker searches for a multimedia service currently registered and capable of solving this goal. Such a multimedia service consists of features combined by device and software capabilities and exhibits supporting sub-services. A detailed introduction to multimedia services is given in Section 5, the matching algorithm is described in Section 6.2.

The sensor tracking module is used to monitor environmental changes. Physical meeting context is included by means of sensors, like temperature or luminance sensors, proximity sensors (for example based on RFID technology), or location systems like the Ekahau positioning engine for indoor location tracking.\(^2\) The generated traces allow the system to react to environmental changes, that is, to become context-aware, to log information for post-meeting processing reasons, and to learn from observations.

Finally, the privacy and security module guarantees access control by means of authorization based on the meeting definition (agenda and goal hierarchy) and a role concept.

### 3.4 Learning & Self-Adaptation

In order to design a satisfactory proactive meeting assistant, the architecture includes a learning and self-adaptation module. Traces stored in the repository represent the lessons learned. In subsequent meetings, the system is able to make use of the digital memory created and can adapt itself to the requirements of the user in a more adequate way.

### 3.5 Object Space

The object space serves as a virtual shared memory for objects that are processed during the meeting. These objects include, but are not restricted to files, persons, devices, and physical objects. For example, a presentation file can be inserted into the object space which will be used by the key-note speaker of a meeting. The objects are assigned to goals and accessed by the matching multimedia services as proposed by the goal/service broker. Service access to objects is restricted by these assignments. The object space, for example, is used to realize chat or file-sharing applications for meeting participants. Object characteristics are described by means of the Resource Description Framework (RDF).\(^3\)

For every object a description based on an OWL ontology is plugged into the system.

---

\(^1\)\(\text{http://www.w3.org/2004/OWL/}\)

\(^2\)\(\text{http://www.ekahau.com/}\)

\(^3\)\(\text{http://www.w3.org/RDF/}\)
4. GOAL MODELING

The goal-driven meeting support approach uses the abstraction of goals in order to separate logical relations of meeting items from the services realizing and fulfilling the goals depending on the physical context of the meeting. This approach is able to support human requirements and preferences in a meeting better because of the following advantages: (1) goals model human preferences and requirements better than services, (2) the specified goals persist although the services might be adapted or changed, (3) the computation of goals and their relations hides the complexity from embedded services in the environment, which are expected to be numerous but simple in terms of processing and reasoning capabilities.4

Based on the implicit and explicit knowledge about human meeting goals, room goals are derived. The following subsections refer to these room goals.

4.1 Goal Representation

We model the goal-driven framework by means of a goal hierarchy and goal relations influenced by first-order logic calculus [14] and the following vocabulary:

**Predicate variables:** $G_i$ denotes a (sub)goal of the set of Goals $G$ which can evaluate a fulfillment degree $f(G_i) : G \mapsto [−1; 1]$ indicating the estimated degree of fulfillment for each goal. For example, 1 indicates that this goal has been optimally reached, 0 indicates that the goal has not been reached, and −1 indicates that a multimedia service has caused malign effects on a goal.

**Goal variables:** $v_i$ denotes a variable used in goal statements.

**Goal constants:** from the goal evaluation perspective, $C_i$ denotes a constant value. However, these constants are defined by the runtime environment in terms of attributes.

**Logical operators:** The logical operators used are defined by $!$ (similar to not), $\&\&$ (similar to and), and $\|\|$ (similar to or).

**Iterator:** The quantifiers are expressed by means of the other logical operators. Additionally, we introduce a new iterator operator $\oplus$, which is an adaptation of the universal quantification in terms of degree calculation.

We include neither conditional nor bi-conditional operators, since these logical operators can be expressed by the other logical operators and have no special merit in our approach. In addition to the possibilities of hierarchical modeling, we use metadata to reason about ordering of goals in terms of time, that is, deadlines. These metadata can be further enhanced, for example, to model fail-over behavior in case a goal cannot be met. Goals do not describe whether they can be executed in parallel or sequential, which is up to the runtime environment (i.e. the broker).

Each goal consists of either a combination of subgoals termed composite goal or a fact termed atomic goal. Leaves of the goal hierarchy, i.e. atomic goals, are mapped to services. No further restrictions of goal decomposition are included in our model. Thus, each goal is denoted by the following n-tuple $G_i = (L_i, C_i, T_i, \ldots, Op_i)$, where $L_i$ denotes the goal’s label, $C_i$ the goal’s attributes, $T_i$ the goal’s deadline derived from the meeting agenda if specified, and $Op_i$ the operator and its arguments in case of a composite goal. Figure 3 depicts a typical goal hierarchy, the corresponding tuples are given as follows:

$$
G_0 : = (L_0, C_0, C_{G_0}, G_{0.0}, G_{0.1})
$$

$$
G_{0.1} : = (L_{0.1}, C_{0.1}, T_{0.1}, \&\& G_{0.1.0}, G_{0.1.1}, G_{0.1.2})
$$

$$
G_{0.1.2} : = (L_{0.1.2}, C_{0.1.2}, T_{0.1.2}, \| (v, C_v, G_{0.1.2.0}))
$$

For a composite goal $G_i$, the degree of fulfillment is calculated according to its goal hierarchy. A weight $a_{ij}$ is assigned to each goal $G_{i,j}$ used for self-adaptation. Depending on the calculated goal fulfillment degree the system suggests multimedia services and monitors user feedback and interactions. In case the recommended services were rated beneficial, the weights will be increased. Otherwise they will be decreased.

For the operations specified, the fulfillment degree is calculated as follows:

$$
\&\& (G_{i,0}, \ldots, G_{i,n}) : f(G_i) = \min_{j=0}^{n} a_{ij} f(G_{i,j})
$$

$$
\| (G_{i,0}, \ldots, G_{i,n}) : f(G_i) = \max_{j=0}^{n} a_{ij} f(G_{i,j})
$$

$$
!(G_{i,0}) : f(G_i) = -G_{i,0}
$$

$$
\oplus (G_{i,0}) : f(G_i) = \sum_{i=1}^{n} a_{ij} f(G_{i,n})
$$

(where $a_{ij} \in [0; 1], \sum_{i=1}^{n} a_{ij} = 1$).

4.2 Goal Ontology

The goal ontology is defined by means of OWL DL. OWL provides superior means for describing both vocabularies and relationships between terms in a machine-interpretable manner. This characteristic is used to describe the previously introduced goal hierarchy. Goals and operators are defined as depicted in Listing 1.

---

4 The embedded meeting services envisioned are expected to be implemented partly by the logic in smart appliances, which exhibit limited memory, processing power, and functionality.
Composite goals (and subgoals) which allow to structure the goals hierarchically are realized via arguments the operator accepts. These arguments can be individuals of the classes Operator or Goal, as is shown in Listing 1.

The operators are all defined in a similar manner. Listing 2 describes how the most complex operator, that is, the iterator, is described in OWL.

5. SERVICES

When acting in an unfamiliar environment it can be obstructive to have a set of rich multimedia devices and no operating experience. In place of the user, the environment can manage these devices and services minimizing the personal administration overhead. In order to access and control a diverse set of components, we propose a service oriented architecture. In principle, this architecture provides an abstraction from the necessary environmental communication. Therefore, the service structure, communication protocols, as well as context features are encapsulated and hidden from the runtime environment. In contrast to using the broker for reasoning about the context (such as proposed in [6]), services aggregate the context parameters by means of the fulfillment degree for a specific goal. Since the broker matches services against goals, the service ontology has to be accessible by the broker. Additionally, the broker transfers information about attributes defined by the goals and bound to the services during runtime. The following subsections detail these concepts.

5.1 Service Architecture

For the design of our system, we choose Web Services5 as interface to components of the service architecture which are depicted in Figure 4. Due to the communication abstraction layer, there is no need for the runtime environment to deal with service specific problems like service discovery or protocols, but the service scheduling and invocation is controlled by the runtime environment. Referring to the goal driven approach all services are considered to be atomic. Service composition is realized through goal composition and matching each atomic goal to one service. Every service accessible by the runtime environment has to provide four functions: (1) estimated degree of fulfillment, (2) estimated time of execution, (3) actual fulfillment after execution is completed, (4) associated service class(es) based on the service ontology.

Listing 1: Goal and operator definition

Listing 2: Iterator operator

Figure 4: Device communication architecture

http://www.w3.org/2002/ws/
We distinguish between two types of service communication protocols. Either a service may be accessed by means of an smart meeting room specific communication protocol, or a standard communication protocol is used. While the first option is beneficial for specific domain dependent services, the latter option allows to include additional services based on well-known frameworks in the field, like Jini\(^6\) or UPnP\(^7\). Standard protocols will be supported in our architecture by means of proxies.

### 5.2 Service Semantics

We choose to support the broker by defining an upper ontology that describes the main concepts of services in the environment. Although OWL-S\(^8\) is currently investigated as a possible standard for the semantic description of web services [16], there are still drawbacks [1]. For the purposes of our goal/service broker we favored a proprietary semantic description of services in OWL DL.

### 5.3 Attributes

The broker supplies the services with goal attributes from the object space based on a standard upper attribute ontology. The services use these attributes not only to calculate fulfillment degrees for goals, but also for service provisioning purposes, like e.g. an MPEG-4 coded video (and its structure) required by a video streaming service. The information transferred from the broker to the service further includes information about the impact to the fulfillment degree each attribute exhibits.

Services use domain specific data structures expressed in RDF to exchange object data based on different standards and formats. Although perfectly suitable for our approach we are planning to verify the MPEG-21 framework [4] as a possible substitution or extension. The benefits of using the MPEG-21 framework as internal attribute representation would be the standardized integration of desired qualities like capability and profile information similar to CC/PP\(^9\), Digital Item Declaration, Intellectual Property Management and Protection, and Digital Item Adaptation.

### 5.4 Capabilities and Profiles

In order to calculate a significant fulfillment degree the service has to be aware of the subsystem, that is, its capabilities and profiles. We assume that services will use standard technologies for this purpose like CC/PP, which—like many other current approaches—only provides syntax descriptions. However, for a seamless integration of diverse devices, networks, and services, it would be important to standardize semantics as well [5]. We propose to hide these semantically heterogeneous capability descriptions by encapsulating these aspects by the service.

### 6. ONTOLOGY-BASED MATCHING

The usefulness of well-defined ontologies, the expression of ontologies using Semantic Web technologies [9], and their usage for matching of requirements and services have been described, e.g. in [13, 6, 10, 8]. In our scenario, we use the power of combining the open world assumption with extensibility. Our architecture only defines a set of upper ontologies, which provide a common vocabulary for the modeling of concrete domain ontologies. The role of these upper ontologies in the matching process is depicted in Figure 6.

![Figure 5: Ontology-based matchmaking](image)

We define only the top-level classes of ontologies to provide a lowest common denominator for the modeling of goals, objects, and services. These upper ontologies are sufficiently expressive. Still, they are sufficiently general so that domain-specific features can be integrated smoothly. For the meeting scenario, an ontology of meeting goals is proposed, which makes use of the common goal ontology. For the definition of meeting objects, there will be ontologies for file objects, person objects, etc. which are dynamically integrated into the system on demand. For services within a multimedia environment, there will be an ontology that describes, e.g. digital white-boards or location sensor technology. Services may describe themselves using the service ontology. However, to keep components small and efficient, they must only implement and understand the parts of the ontologies required for their functionality. The following subsections describe in detail how goal attributes are bound to values and how matching services are found based on these ontologies.

#### 6.1 Attribute Binding

As described in Section 4, attributes are used to describe goals in more detail. Before a goal can be matched to a service, all its attributes must be bound to objects residing in the object space. As described above, every object in the object space is described by a metadata record. During the definition of the agenda, the meeting moderator defines a set of matching criteria for goal attributes. For example, for a variable describing a Powerpoint presentation by Niko Popitsch, the matching criteria may be \{$\text{dc:type} = \text{".ppt"}, \text{dc:title} = \text{"METIS in a multimedia environment"}, \text{dc:creator} = \text{"niko.popitsch@researchstudios.at"}$\}. If no such objects are available, or the appropriate object cannot be determined unambiguously, the meeting moderator is prompted to provide the object, or to select one out of the set of appropriate objects.

#### 6.2 Goal Matching

All goals whose attributes are bound to objects are organized into a goal priority list. Accordingly, atomic goals are matched to available services. The result of this processing is a list of services \(E_G\) that are able to fulfill goal \(G\) with a degree of at least \(q\), ordered by their estimated execution time.
In detail, the algorithm for matching a set of goals $A_G$ to a set of available services $S$ is depicted below. Each service $S \in S$ must provide a function $S.fulfills(\cdot) : A_G \mapsto [-1; 1]$, indicating the estimated degree of fulfillment for each atomic goal, where 1 indicates perfect support for this goal, 0 indicates that the service has no effect on this goal or is not aware of the goal’s description, and $-1$ indicates that the service has a destructive effect on this goal. Furthermore, each service must provide a function $S.time(\cdot) : A_G \mapsto \mathbb{N}$, which indicates the amount of time that this service requires to process the request. This function is necessary for timely invocation of services. In order to increase efficiency, services may describe themselves in terms of classes of the service ontology: $S.classes(\cdot) : \{\} \mapsto SC^*$, with $SC$ being the set of service classes. Using this function, the broker is able to restrain the set of services to be queried to those belonging to specific classes, namely the classes that were able to fulfill the goal in previous situations.

The algorithm can be configured using a quality threshold $q$. If possible, only services with $S.fulfills() > q$ are selected. If none of the services can provide an adequate degree of fulfillment, the matching returns an empty set. Subsequently, the user is asked to reduce $q$ so that a matching service can be found.

**Algorithm 1 Match($G, S, q$)**

$B_G \leftarrow \{\}$

for all $S \in S$ do
  if $\exists S.classes() \in \text{classes}(G)$ and $S.fulfills(G) \geq q$ then
    add $S$ to $B_G$
  end if
end for

order elements $S \in B_G$ according to $S.time(G)$

In a second step, the list of eligible services is further processed, considering the goal trees for the situation. In particular, goals that are outperformed by their alternatives within one |($) term are removed from the list. Goals that are argument to an iterator ($\exists$) must be multiplied so that the corresponding services repeatedly occur in $B_G$, identified by different parameter instances. If there are fulfillment conflicts (conflicts that occur because one capability supports one goal, but obstructs another one), they must be resolved so that the overall goal fulfillment is maximized.

Finally, if there is only one service in $B_G$, it is scheduled for execution. Otherwise, the meeting moderator may select one service that is appropriate for him, or the goal/service broker may select a service based on global configuration options. Moreover, the self-adaptation module might help at this point. Preceding user selections and interactions will be analyzed to present alternatives.

### 6.3 Meeting Processing

The first cycle of goal/services matching is executed at the beginning of the meeting. For every scheduled goal/service relation, there exists an optional start time of execution. This value is calculated from the estimated time that a service requires to fulfill a goal and the time that the agenda schedules for this goal to be fulfilled. Thus, the goal/service broker is able to invoke services timely.

However, the execution schedule must be revised by the goal/service broker regularly because of various reasons. The execution of a service may fail, e.g. because of technical problems. In this case, the corresponding goal must be reassigned to the second best service which is able to fulfill it. Additionally, the goal hierarchy may change during the meeting because of short-term modifications of the agenda. Then, the new or modified goals must be (re)matched to services, or cancelled goals must be removed from the execution schedule. Finally, the service environment may change due to context dynamics. New services may register themselves (in this case, the goal/service matching must be revised to check whether there exist new services that may fulfill goals better), or services may disappear from the system. Parameters for goal fulfillment regarding the meeting context (e.g. the persons which are present in the room, the lighting conditions, or the temperature) may change; hence, goals that were assigned to services must probably be re-matched to other ones.

### 7. APPLICATION SCENARIO

To give an impression how a real meeting (Figure 6) in a multimedia environment is conducted, the following textual description of an application scenario illustrates the interaction between meeting attendees and the system as well as the collaboration of the system parts. This use case is presented in a descriptive manner from a user’s point of view, yet still mentioning system internal processes triggered by user interaction. (A formal use case description is omitted, because it would exceed the size of this paper.)

#### 7.1 Premeeeting Phase

In the premeeeting phase the person responsible for arranging and preparing the meeting defines the agenda of the meeting. This is realized by an user interface of the agenda preparation module. It provides the possibility to enter parameters like the list of participants ($P_{1...5}$) and the agenda items ($A_{1...5}$). The agenda items for this sample meeting are:
• (A1) Welcome message
• (A2) Presentation of project 'Sample Project' by P1
• (A3) Discussion
• (A4) Coffee break
• (A5) Collaborative editing of 'Sample Document'

The agenda items are internally represented as instances of classes of the agenda ontology. The ontology itself is formulated in OWL DL, as well as the other ontologies in the system (i.e. goal ontology, service ontology, attribute ontology). For essential fragments of the sample meeting’s representation see Listing 3, containing the instances of the meeting, the moderator, a remotely located participant and two agenda items. Omitting URIs in favor of short names as identifiers in the following sample listings is intentional to make them more readable.

Listing 3: The agenda’s representation depicting the sample meeting in abbreviated form

```
...<Meeting rdf:ID="Sample_Meeting">
    <moderator>
        <Person rdf:ID="P1">
            <name>name_P1</name>
            <location>
                <local rdf:ID="Meeting_R">
                    <attends rdf:resource="#Sample_Meeting"/>
                </local>
            </location>
        </Person>
    </moderator>
    <participant>
        <Person rdf:ID="P2">
            <remote rdf:ID="Office_P2"/>
            <location>
                <attends rdf:resource="#Sample_Meeting"/>
                <name>name_P2</name>
            </location>
        </Person>
    </participant>
    ...<agenda_item>
        <Presentation rdf:ID="my_Presentation">
            <topic>Project XYZ</topic>
            <duration>30</duration>
        </Presentation>
    </agenda_item>
    <agenda_item>
        <Break rdf:ID="my_Coffee_Break">
            <topic>none</topic>
            <duration>15</duration>
        </Break>
    </agenda_item>
    ...</Meeting>
```

Through analyzing the list of participants and their contact information known to the system, participants are informed about the meeting (e.g. for participant P2 an entry in her Outlook calendar is placed, for participant P3 a notification via email is sent...). Additionally, required resources are reserved (e.g. meeting room R on date d from t_b to t_e...).

If the user authorizes the agenda, it is transformed into a goal hierarchy by the goal definition module. The goal hierarchy is created based on the predefined goal hierarchies related to a certain kind of agenda item. In the first stage a full hierarchy is built, which is further manipulated through the learning & self adaption module in a second step. However, branches the user has marked as non-relevant in goal hierarchies of former meetings are omitted. Finally, if necessary, the goal hierarchy can be manipulated directly by the user herself. In Listing 4 the essential fragments of the goal hierarchies’ representation depicting the agenda item “Presentation of Project ‘Sample Project’” are shown. It is assumed that this goal representation has already passed all stages of post-processing mentioned before. The listing contains a branch of the goal hierarchy describing the goal to convey presentation Q to the meeting participants P_i. This goal G_0 should be fulfilled by the subgoal G_1 (i.e. visualize the presentation) and G_2 (i.e. deliver a copy of the slides). Goal G_1 can be fulfilled by its subgoals, either G_10 (i.e., visualize the presentation via a shared output device), or G_11 (i.e., visualize the presentation via an individual output device iterated over the list of participants).

Listing 4: The goal hierarchies’ representation depicting the agenda item “Presentation of Project ‘Sample Project’” in abbreviated form

```
...<Goal rdf:ID="G0">
    <variable rdf:resource="#var_Presentation"/>
    <variable rdf:resource="#var_Participants"/>
    <label>convey presentation Q to participants P_i</label>
    <composite>
        <And rdf:ID="And_G0">
            <argument rdf:resource="#G1"/>
            <goal rdf:ID="G2">
                ...
            </goal>
        </And>
    </composite>
    <is_part_of rdf:resource="#Sample_Meeting"/>
</Goal>
    ...<Goal rdf:ID="G1">
        <composite>
            <Or rdf:ID="Or_G1">
                <argument rdf:resource="#It_G11"/>
                <argument>
                    <Goal rdf:ID="G10">
                        <label>display presentation Q on shared visual output device</label>
                        <variable rdf:resource="#var_Participants"/>
                        <is_part_of rdf:resource="#Sample_Meeting"/>
                        <variable rdf:resource="#var_Presentation"/>
                    </goal>
                    <has_argument/>
                </argument>
                <composite>
                    <variable rdf:resource="#var_Presentation"/>
                    <variable rdf:resource="#var_Participants"/>
                    <label>display Presentation Q to Participants P_i</label>
                </composite>
            </Or>
        </composite>
    </is_part_of rdf:resource="#Sample_Meeting"/>
</Goal>
    <Iterator rdf:ID="It_G11">
        <argument>
            <Goal rdf:ID="G11">
                <label>display presentation Q on personal output device</label>
                <is_part_of rdf:resource="#Sample_Meeting"/>
            </goal>
            <variable rdf:ID="var_Participants"/>
            <variable rdf:ID="var_Presentation"/>
        </argument>
    </Iterator>
```

10The branch describing G_2 was stripped out of the listing because of lack of space.
7.2 Conducting the Meeting

The main responsibility for a successful conduction of the meeting is taken by the goals/services broker. In a first step the variables of goals are bound to their instances (e.g. for \( G_1 \) the variables are the participants \( P_i \) and the presentation \( Q \) itself). Subsequently, according to the deadline parameter of the goals, they are sorted into a goal priority list and further processed by the broker. Considering goal \( G_{10} \) (visualize the presentation via a shared output device) as an example, the service \( S_{23} \), which is capable of displaying the presentation \( Q \) in its format on a beamer, returns an estimated fulfillment degree of 0.9, while the service \( S_{24} \) returns a degree value of 0.01, because it has only access to a very small output screen (e.g. smart-phone display). Following the algorithm defined in Section 6.2, the broker would decide for service \( S_{23} \) to fulfill goal \( G_{10} \). After the presentation was effectively displayed the service returns the actual fulfillment degree back to the broker. See Listing 5 for a service description.

Listing 5: A service description in abbreviated form

```xml
<Device rdf:ID="Monitor_42">
  <Device_service rdf:resource="#S23"/>
  <service_function rdf:resource="#S23"/>
</Device>
```

This procedure is executed for all atomic goals of the agenda items the meeting consists of. Hence, the successful execution of the meeting is ensured. The recorded data about the actual fulfillment degree of goals is stored in the system’s repository and used as input data for the learning & self-adaptation module and for evaluating the performance of the system for the conducted meeting.

8. CONCLUSION

In this paper, we presented an approach to model the user requirements in multimedia-enriched environments based on the concept of goals. We introduced a goal-driven service-oriented architecture for the integration of simple multimedia services and applied the approach to the use case of a smart meeting room. The proposed software service architecture is based on the separation of goals and multimedia services, which reduces complexity for services and increases flexibility. We described how Semantic Web-based ontologies can be used for the modeling of such goals and services, and how the services are matched to goals in order to fulfill user requirements while the context changes dynamically. Finally, we demonstrated our concept for a multimedia-enriched meeting scenario. In future work, we plan to exploit our architecture for new multimedia services and self-adaptive goal composition.

9. ACKNOWLEDGMENTS

We thank Werner Winiewater, Niko Popitsch and Bernhard Haslhofer for their valuable comments and suggestions.

10. REFERENCES


