SEAMLESS INTEGRATION OF OUTPUT DEVICES IN INTELLIGENT ENVIRONMENTS: INFRASTRUCTURE, STRATEGIES AND IMPLEMENTATION

Y. Ding, C. Elting, U. Scholz
European Media Laboratory GmbH, Germany
{yun.ding, christian.elting, ulrich.scholz}@eml-d.villa-bosch.de

ABSTRACT

We present a novel software infrastructure for the seamless integration of heterogeneous output devices into intelligent environments. It dynamically co-ordinates the output devices to enable adaptive, multi-media presentations by taking the availability of resources and the context of the presentation into account. In this paper we focus on three topics. We present our connecting middleware which supports ad-hoc spontaneous interactions between heterogeneous devices connected by different types of networks. We focus on the multimedia output (MMO) channel which is situated atop this middleware and which is responsible for coordinating output devices according to their self-descriptions. Then we give details about the AI planner which is used by the MMO channel to compute a solution to the presentation problem given by the presentation context. Finally, we outline our implementations for an intelligent living room scenario and point out related work.

1. INTRODUCTION

John is sitting in his living room and wants to watch some movie on TV tonight, but he does not know which one. Therefore John picks up the PDA next to him, with which he can control his intelligent living room. With the PDA John is able to request information about the movie "Rio Bravo" which is shown tonight. He receives images and a short textual synopsis of the movie which are accompanied by an animated character (see the left part of Figure 1). However John wants to view the images on a bigger screen. Therefore he switches the images to the TV set in the living room and only the synopsis is displayed on his PDA. A digital picture frame nearby is used to display another image from the movie (see the right part of Figure 1). John wants to get some juice and peanuts from the kitchen. In the kitchen the system alerts John that "Rio Bravo" is about to begin and if he wants to record or to watch it. This is done by means of a synthesized voice over the speakers in the kitchen as John is in no direct view of a display. Being properly equipped with food and juices John sits down and begins watching the movie.

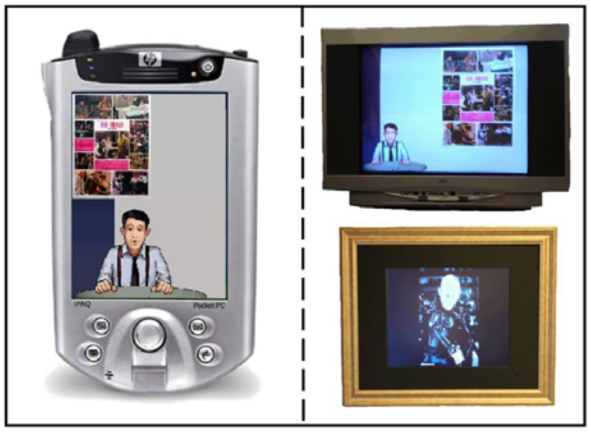

Figure 1: Movie information on a PDA respectively on a TV set and a digital picture frame

From this scenario we can derive some requirements on an infrastructure for the coordination of output devices in intelligent environments:

- Ad-hoc spontaneous connectivity: Heterogeneous devices and intelligent appliances from different vendors using different standards interact without any need of pre-configuration. For instance, John's PDA seamlessly interacts with the TV set and the digital picture frame.
- Adaptive user interfaces and presentations: The currently active output device changes from John's PDA, the TV set and the digital picture frame to the speakers in the kitchen once he leaves the living room. A seamless migration of user interfaces (UIs) and presentations to different devices which significantly differ in their resources, requires UIs and presentations to be resource-adaptive in order to be rendered on different devices.
- Multi-device presentations: Intelligent interaction with distributed output devices within dynamically changeable ensembles requires software technology that makes coherent acting of output devices possible. The presentation must be decomposed into parts which can be rendered on the single devices, and eventually the devices must be synchronized.

In this paper, we propose a novel software infrastructure which supports an adaptive multi-device output generation independent of underlying physical connections. The architecture and the API defined by our infrastructure allow devices from independent vendors to spontaneously discover and interact with each other in a collaborative and coherent way. We first describe the big picture, a general architecture which tackles the above requirements in multiple layers. Then we describe the MMO channel which coordinates multiple output devices by means of self-descriptions. Afterwards we describe the MMO presentation planner which is the core of the MMO channel. Finally we describe our implementation as part of an intelligent living room and conclude with an overview over related work.

2. THE BIG PICTURE: A LAYERED ARCHITECTURE

The scenario described above is typical in ad-hoc, peer-to-peer (P2P) computing. P2P systems are distributed systems based on the concept of resource sharing by direct exchange between peers which have the same role and responsibility. The environments in which P2P applications are deployed are extremely dynamic in structure, content and load. The topology of the system typically changes rapidly, either intentionally, due to peers entering and leaving, or coincidently, due to events like crashes and network failures. In our context, a peer can be a device, a component of a device or a piece of software. We use a layered architecture (see Figure 2) to meet the above requirements:

- The *Physical Connection Layer* represents the real physical network consisting of a diversity of (incompatible, non-interoperable) networking protocols and technologies.
- The *Abstract Connection Layer* (ACL) establishes a virtual network on top of the physical network allowing peers to discover each other dynamically, interact directly, and organize into groups independently of their network location (e.g., firewalls or NATs).
- The *Coordination Layer* supports the self-organization, a sort of distributed coordination, of peers. The dynamic environment as such excludes any form of central control. Neither the knowledge of other peers nor their presence at a given time can be a priori known or planned. The peers interact and coordinate themselves through the space/environment where they are located in. In this paper, we focus particularly on the MMO channel which serves as a distributed, intelligent environment which coordinates the output devices. We adopt the terminology of the SodaPop model [11]. A channel offers a source and a sink end to the components. It receives messages from its source and passes them to the components on its sink end according to the „policy of the channel" which defines the way to process and distribute incoming messages to components on its sink end. The MMO channel is part of this infrastructure.

For the sake of readability Figure 2 provides only a simplified view:

- The circles in Figure 2 are not necessarily stand-alone devices. They can be internal components of a device, or system software components like a dialogue manager. For example, a TV set consists of a remote control, a screen and a loudspeaker. They are represented as three circles in the architecture respectively.
- The interaction is not strictly downwards as depicted in Figure 2. Upwards interactions among the components are necessary, for example, to allow actuator components to give feedback on the outcome of an action.

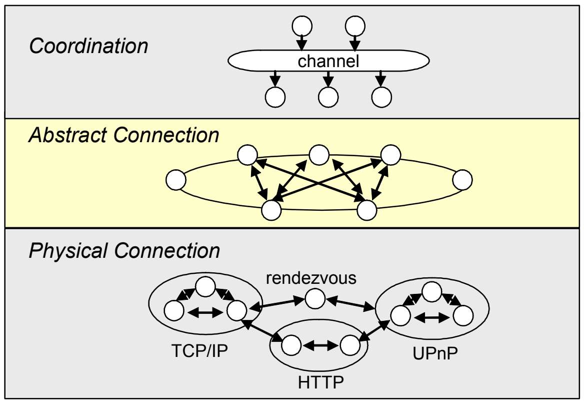

Figure 2: The layered architecture

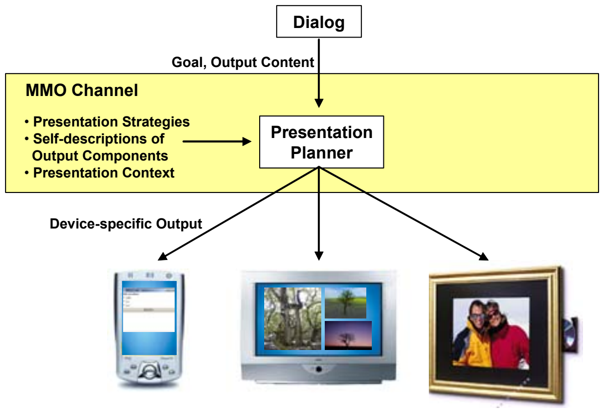

Figure 3: The MMO channel

Figure 3 depicts the MMO channel which is located in the coordination layer in our architecture in more detail. Upon receiving an output request consisting of an abstract output goal and the content to be presented, the MMO channel uses an AI planner to generate a list of output actions which are to be executed by single output components. The planning process depends on the presentation strategies and contextual information such as the self-descriptions of the output components. The planner translates the actions into appropriate languages and sends them to the output components. Section 4 and Section 5 describes the MMO channel and the MMO presentation planner in more detail.

3. THE CONNECTING MIDDLEWARE

Before any form of self-organization can take place all the different devices in the networks have to be connected. The Abstract Connection Layer (ACL) acts as a tie between the underlying physical networking technologies and the coordination layer. It encapsulates the details how devices are discovered, connected and interact as well as how channels are realized. The services of ACL can be grouped into three categories: discovery, group-formation and peer communication:

- *Discovery*: Whenever a peer enters the network, it registers with the local ACL. The ACL creates a peer description, publishes it on the network, searches for existing peers in the network, and polls for incoming new peers. In this manner, the ACL announces the existence of the local new peer, discovers other peers in the network, and is able to notify the local peer of the result of the discovery.
- *Group-formation*: The peers connected to the same channel form a group. The ACL allows peers to dynamically create, join and leave groups. In the same way of peer discovery as described above, ACL discovers incoming and leaving group members and informs the existing group members.
- *Peer communication*: The peers either interact directly or anonymously via the channels. The latter is a kind of group communication. In case a message is addressed to a channel, it is the task of the channel to distribute it to the interested peers according to its policy.

E.g., a self-description of a TV set is published onto the home network and should be accessible by other components of the system. John's PDA, his TV set, the picture frame as well as the speaker in the kitchen are connected to the MMO channel. They build up a group to collaboratively render information needed by their owner. Whenever a new output device is bought into the network and connected to the MMO channel, it will be detected and taken into account for future rendering tasks.

The infrastructure for dynamic, intelligent environments should not itself build a central point of failure. Therefore the middleware is physically distributed among the devices in the network. Each device hosts one instance of the middleware if it has sufficient resources. Resource-constrained devices which are not able to host the middleware by themselves are connected to a remote middleware instance hosted by a more powerful device via proxies. Being a constituent of the infrastructure, the MMO channel is also distributed over separate devices. It (or more precisely, its first instance) will be dynamically created by the first component which joins it. The ACL announces its description such that other components get known of it and can subsequently join it by creating a new instance. From the point of view of the ACL, each instance of the distributed middleware can be considered as a peer. Thus, the ACL also supports the peer-to-peer communication between the middleware instances beyond the device boundaries.

The JXTA project [13] and the Universal Plug and Play (UPnP) forum [18] have defined protocols which are concerned with some of these services (e.g., discovery and interaction) provided by the ACL. The UPnP technology has been widely adopted by the entertainment industry. Its nature of zero-configuration makes it efficient and easy-to-use. However, compared to UPnP, the JXTA-protocols offer a higher level of abstraction, network independency and scalability. For instance, JXTA can be used to interconnect disparate UPnP networks that are separated by firewalls.

Although the implementation of ACL is currently based both on JXTA and on UPnP (see Figure 4), it is beyond them. In UPnP, devices have unequal roles. The so-called control points are capable of controlling other devices. To enable peer-to-peer networking, each peer considered by our middleware is modeled as a device which incorporates a control point functionality. Additionally, we had to implement a group concept which is not supported by UPnP.

In contrast to UPnP, JXTA is designed to be able to support real large-scale P2P systems. Its discovery service uses a "publish and search" pattern [6], meaning that it can only discover descriptions which it is told to look for. There is no way to just *listen* for published descriptions of peers. The reason is that a peer would otherwise be overwhelmed by descriptions from the numerous peers of the large, open system. Therefore, we utilized additional JXTA services (for example, the rendezvous service) to implement the discovery service of ACL as described above.

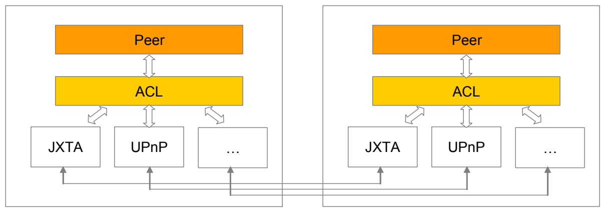

Figure 4: P2P communication supported by ACL

4. MMO CHANNEL

The MMO channel is the part of the middleware which is responsible for adapting, distributing and coordinating the content to be presented between multiple output devices and components (Figure 3). This is done by means of the presentation planner. The MMO channel uses the ACL as described in Section 3 to access output devices and components independent of their network protocols. In this section we first identify the requirements for MMO presentation planning and then illustrate the approach taken. Then we give details about our model of self-descriptions which cover IO devices, services and resources. After that we illustrate our presentation strategies. The presentation planner is described in Section 5. It uses the presentation strategies to generate a multi-device layout depending on the output task, the available IO devices, the user preferences, and the current context.

4.1 Requirements

In this Section we first analyze the requirements for the presentation planning process. Figure 5 summarizes these requirements.

In the example from Section 1 the output of the movie scenes is directed to John's PDA. This is done by watching John's interactions and inferring that John used the PDA for input and therefore will expect output to be located on the same device. The interactions are part of the *dialogue context*. The dialogue context also includes the history of the output generated by the system. This serves to harmonize the layout of new presentations with that of former presentations.

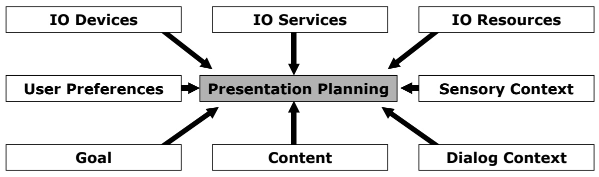

Figure 5: Presentation context

When John and Mary watch the movie scenes together a combination of a TV set and a digital picture frame is used to present the output. Here the system needs to know about the *resources* on both devices. The TV set has a 640x480 resolution as a difference to 1024x768 on the digital picture frame. Apart from screen resolution, resources also cover computing power (e.g., given as networking or graphics benchmarks) as well as the possible volume settings of acoustic output devices. For the MMO channel it is also important to know about the available output software or *services* on each device. Some devices might be able to render ASCII text into speech whereas others are only able to display it graphically. Moreover the user should be able to choose from different layouts on both devices. Therefore a *user preference* model is needed.

Input devices need also to be modeled for coordinating output devices, because the device chosen for output has to fit the input device the user has used, e.g., John's PDA in our example.

In the final example from Section 1 the speakers in the kitchen are used by a speech synthesis to alert John that his movie is about to start. For this scenario the system needs information about the position of the user within the room, in order to infer to use speech instead of a graphic presentation. Moreover the system needs to know the position of each output device within the room. We summarize this information as sensory context.

When the system alerts John in the kitchen by means of acoustic output, it could also have displayed graphic output, e.g., a silent alarm on the TV screen. However due to the importance related to the content of the message the system chose to use an output format which immediately reaches John perception. This shows that the *goal* of the output is important for the generation of presentations. The same is true for the type of the output *content* to be displayed. E.g., geographical data can be visualized very well graphically whereas it is often difficult to render it acoustically.

We call all the factors that influence the generation of a presentation the *presentation context*. In the following sections we first illustrate our approach to meet these requirements with the MMO channel. Then we illustrate our model for IO devices, services and resources. Afterwards we give examples for presentation strategies which use this model. In Section 5 we then give details about the presentation planner which is the core of the MMO channel and uses the presentation strategies to coordinate the available output devices according to the presentation context.

4.2 Our Approach

The general idea of the MMO presentation generation process is to let a presentation planner perform the adaptations based on the presentation strategies and on the presentation context (see Figure 6). In other words, we do not design one hard-wired layout for each device or even for each device combination.

The self-descriptions include information about the available resources and the services of input and output devices. Resources include, for example, the screen resolution of the device and its networking and graphics benchmarks. This makes it possible to adapt images to the size of the PDA screen and to adapt audio quality to the networking resources of an output device. Apart from the output resources also information about the output services located on this device are needed. On one output device only HTML content may be rendered, while on another device a speech synthesis may be present.

The content of the presentation includes information about the media objects available for presentation. These include ASCII text, images, audio files or video files. The goal of the presentation is also modeled.

Possible goals include general information of the user as opposed to displaying an error message.

This information (together with user preferences, sensory context and dialogue context which are not displayed here) makes up the presentation problem. It is inserted into the presentation planner (see Figure 3) which is described in detail in Section 5.

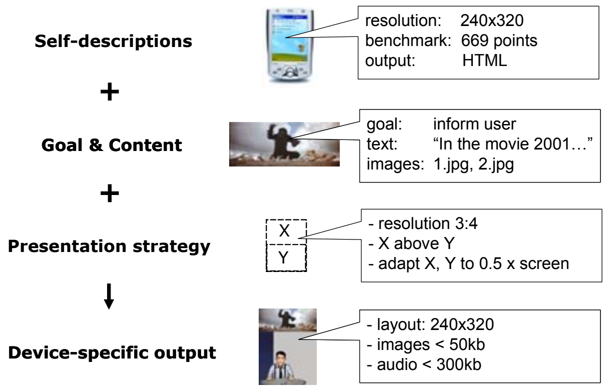

Figure 6: Adaptive output generation by the MMO channel

To solve the presentation problem the presentation planner uses a set of presentation strategies. It is important to note that these strategies are defined device-independently, e.g., by means of constraints. Instead of saying that one image should be located at position (X,Y) and another at position (X', Y') the strategy only says, that the first image should for instance be located above the second one without restricting the strategy to a certain screen resolution.

This makes it possible to use only few strategies to support a large set of possible output devices which can all be connected and disconnected at runtime. As a result the device-specific output is generated by the presentation planner and sent to the device as a result of the MMO output coordination.

4.3 Modeling IO Devices, Services and Resources

Next we describe our model of IO devices, services and resources which makes it possible for IO devices to self-identify. A first version of this model appeared in [8].

First every IO device is classified as being either graphic or acoustic. We provide the IP address of the physical device as a unique identifier of the physical device of which the IO device may be part of (Table 1). Therefore a PC might contain a graphic output device (i.e., a screen) and an acoustic output device (i.e., speakers) which both have the same ID. An example for a graphic input device is a mouse. Here we do not model the actual input device in detail (i.e., we do not distinguish between track pad, track ball and mouse). With an acoustic input device we model a microphone.

It is important to know about these devices, because if there is no graphic input device then the user is not able to give mouse input at all and presenting a button to press might not be appropriate. For each graphic output device we give information about the current screen resolution of the device. Further resources like networking or graphic benchmarks are not yet taken into account.

A component which offers an input service is called *input component*. These input components are modeled by means of Bernsen's taxonomy of unimodal input [4] which makes it possible to distinguish between a large set of different input techniques. Additionally we provide information about the current state of the input device, including if this input device was selected for input. E.g., if the user uses speech recognition for input then speech might also be used for output. We also include a reference to the input device on which the input event has been processed.

Input	Output
-	multimodal type
unimodality	Unimodalities
-	generated media
-	processed content
state	State
device	Device

TABLE 1: Self-description of IO components

Analogous to input devices we call the components which offer output services *output components*. Output components are described by means of Bernsen's taxonomy of unimodal output [4]. However for output components this is not expressive enough. An output unimodality is defined as only a basic output part, e.g., text output or speech output. Many output components consist of more than just one output unimodality, e.g., a video consisting of animated images and speech. Therefore we describe output components by means of a set of unimodalities and not as a single one. Then we also keep track of the content which can be processed by the output component. This might be SMIL in case of a SMIL playing component or ASCII-Text in case of a text-to-speech engine. For components which only generate output and not necessarily display it, we also provide information about output media which are generated. An example for such a component is a map renderer which receives geographical data and returns a map without displaying it. Similar to input devices we also provide information about the current state of the output components. For graphic output components this includes information about the visibility and possibly the screen coordinates and size of the window in which the output component is displayed. For dynamic output components we keep track if the component is currently playing, paused or was stopped.

However this information is not enough to distinguish between all possible output components. This is because up to now we only describe the parts of which the output component is composed of, but not its semantics. Therefore we include additional information for each output component which serves to describe the output component as a whole. We call this information *multimodal type*. We currently model two multimodal types: players and agents. Output components of type agent are output components which realize parts of a (possibly animated) assistant which serves as the system's anthropomorphic representation for the user. Agents include speech syntheses or the graphics-only part of an animated character. Output components of type player are components which only display a given content without adding new semantic information. Examples for players are a SMIL player or an AVI player.

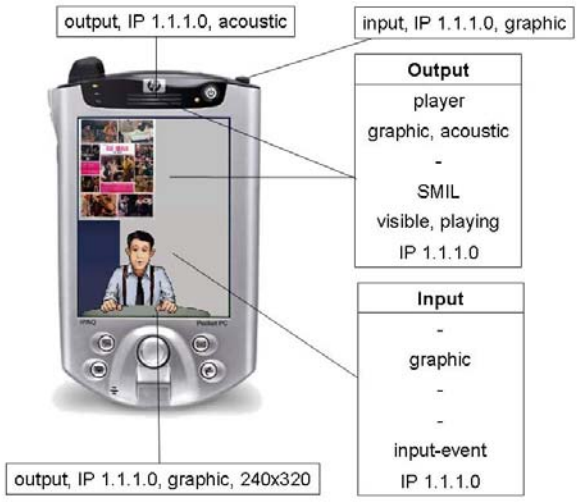

Figure 7: Self-description of input and output components on a PDA

Figure 7 shows an example of a PDA including pen-based input and a SMIL renderer (haptic keyboard input is omitted in this example). The device is composed of two output devices and one input device. The acoustic output device refers to the PDA's speakers. The graphic output device describes the screen of the PDA including as a resource restriction its resolution. The graphic input device refers to the pen-based input on the touch screen of the PDA. The graphic input from the pen is being processed by the input component which processes events from the touch screen. The input component is able to pop up a quick menu which allows the PDA user to select between different controls of the intelligent living room. Currently an input event was processed by this component which means that the last input of the user was done using the PDA. The output component on the PDA renders output on the PDA screen as well as on the PDA speakers. It does not generate any output media, it can process SMIL content and it is currently visible and playing a SMIL presentation. The content of the presentation consists of an animated character. However this character is not modeled as part of the SMIL output component. Its self-description is included in the output component which provided the animated GIFs which are part of the SMIL presentation. This output component is not displayed in Figure 7.

4.4 Presentation Strategies

Presentation strategies are used to describe how the content can be distributed among the available output devices. To do so they use the self-descriptions as defined in Section 4.3 together with the presentation context (Figure 5). Moreover a constraint solving approach guarantees independence from screen resolutions. More details about presentation strategies can be found in [10].

5. MMO PRESENTATION PLANNER

The *presentation planner* has to satisfy requirements from various sources: It has to take into account a proper presentation strategy, the current state of the device ensemble, the requested output and the current state of an ongoing presentation. Furthermore, a presentation has to appear as a continuous process while at the same time it has to react to changes of the ensemble and to new output tasks. The *presentation problem* is the combination of all these requirements and its solution has all of them satisfied.

5.1 The Presentation Problem

At its core, the presentation problem is an Artificial Intelligence (AI) *planning problem*. In short, AI planning is concerned with changing a world from its current state into a state with desired properties. A planning problem consists of two parts: a *planning domain*, roughly the description of the world, and a pair of an *initial state* of the world and a *goal*. The domain includes of a set of *actions*, i.e., transformations whose executions change the world into a new state. A *plan* is a list of actions, that can, one after the other, consistently change the initial world state into a state satisfying the goal. A program, that solves planning problems, i.e., that finds plans for a given problem is called a *planning system*.

The *core presentation problem* is an AI planning problem which is composed of various causal dependencies, temporal and spatial constraints resulting from the current output requests. In our scenario, John wants information about a movie, preferably on a screen. Consequently, the system has to present all available information about the requested movie, and not about other movies (causal dependencies), on an output component with a screen which is large enough (spatial constraints) before it waits for a user reaction (temporal constraints). The final presentation for John is a list of detailed instructions that, executed as a sequence, specify for each output component which media is rendered when and at what coordinate, in other words: a *plan*.

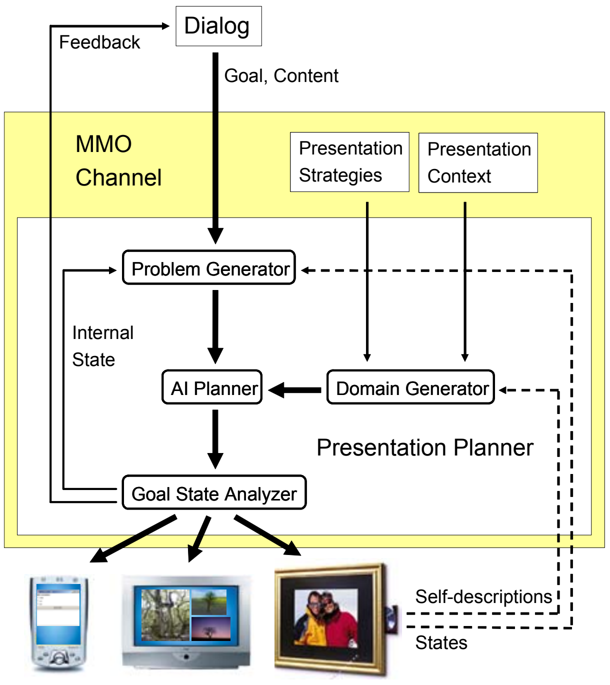

Figure 8: The presentation planner

There is more to a presentation than just being a list of instructions for devices to present media: Presentations have a hierarchical structure that governs their appearance and their development over time. Consider, for example, the movie information, that John requested. This information might contain a set of images, which have to be rendered one at a time. The first image can be shown as soon as the screen is unoccupied, and whilst it is being displayed the other images may not be displayed. Each subsequent image can be shown if (a) it is from the movie, (b) it has not been shown previously, (c) all preceding ones have been shown (let us assume they are ordered) and (d) the screen is unoccupied. These requirements correspond to a recursive list-like decomposition of the task.

Hierarchical Task Network (HTN) *Planners* are AI planning systems that are based on *tasks* and their decomposition into *subtasks*, and hence are suitable for writing and solving presentation planning problems. A task is an endeavor that has to be performed, for example rendering an image on a screen and, more generally, creating a presentation. Each task can be solved in different ways, meaning that it can be decomposed into different set of subtasks. The way how it is decomposed is called a *method*. Therefore, each task is associated with a set of possible methods. A method has attached preconditions which have to be satisfied in case it is employed; they encode causal dependencies between different parts of a plan. A task is considered to be solved once it is decomposed into primitive, indecomposable subtasks (which are referred to as *actions*) and their respective preconditions are satisfied.

5.2 Presentation Strategies and Self-Descriptions

The domain of the core presentation problem is composed of three parts: The first one consists of presentation strategies which define the general manner of presentations, i.e., the range of proper kinds of presentations the system is capable of generating (presentation strategies are detailed in Section 4.4). The second part consists of the presentation context which contains self-descriptions of the devices. These self-descriptions specify the abilities and restrictions of the currently available devices. The combination of these two parts allows for realizing a presentation that follows a general outline with a concrete device ensemble.

A self-description of an output device includes action schemata which allow the presentation planner to directly control the device. Each action comes with an associated sequence of control schemata which are native to the issuing device. The planner maps each instantiated action of a solution to the corresponding control sequence and sends it to the respective device. In short, the use of an action in a presentation directly results in the desired behavior of the device.

For example, consider the problem of rendering a picture on a screen. The device understands and can directly execute SMIL, i.e., a language to describe the temporal, positional and interactive behavior of a multimedia presentation. The presentation planner uses a different input language which is suited for planning with tasks and methods. Both representations are brought together by the self-descriptions of the device: It provides a pair which consists of the action display(device,media,x,y,w,h,t1,t2) and the SMIL schemata <region id="reg1" fit="meet" left="x" top="y" width="w" height="h"/> and <img region="reg1" begin="t1" end="t2" src="media"/>, where the first specifies a region on the screen of the device and the second renders a picture in that region. After the planning process the presentation planner collects all planning instructions for each device. From each such collection it produces a list of SMIL instructions by instantiating the SMIL schemata in the same way that their corresponding planning action is. Finally, the SMIL instructions are combined to a valid SMIL script and sent to the originating device.

In our example let us assume that the planning action is part of a presentation and that it is instantiated to the action display(screen1,"http://172.16.3.21/3.jpg",20, 30,10,15,5,27) which means that the device screen1 renders the specified picture at the xy-coordinates

20/30, that the picture has a width of 10 and a height of 15 and that its presentation starts at time point 5 and ends at time point 27. This action results in the SMIL instructions <img id="reg1" begin="5" end="27" region="screen1" src="http://172.16.3.21/3.jpg"/> and <region id="reg1" fit="meet" left="20" top="30" width="10" height="15"/>. The presentation planner combines these instructions, together with other instructions for device screen1, to a valid SMIL script which is sent to the specified device.

The presentation planner solves the core presentation problem by bringing together presentation strategies and presentation context. The root task of the presentation planner is called *plan_presentation* and represents all possible presentations. Presentation strategies are modeled by high-level methods. The result of presentation planning is a decomposition of *plan_presentation* into actions which satisfy the current output requests with the current device set. This plan realizes a coherent presentation for a given set of causal, spatial and temporal dependencies.

5.3 Components of the Presentation Planner

An AI planning system is able to solve one instance of a core presentation problem at a time. But on its own, it does not satisfy the additional requirements on presentations. These properties are realized by components which, together with the AI planning system, comprise the presentation planner (see Figure 8). Those requirements are *reactivity*, *adaptivity* and *continuity*.

Adaptivity means that the presentation planner knows about the available devices and uses them in the best possible way. A static table of devices descriptions cannot fully support this ability because output devices are highly heterogeneous and not all possible descriptions can be stored by the system a priory. Instead, the presentation planner adapts to a specific situation by maintaining a knowledge base of the available set of devices. Once, at the time a device enters the system, it sends a self-description about its capabilities, restrictions and its current state to the presentation planner (cf. the dotted lines in Figure 8). From these self-descriptions, together with the presentation strategies (see Section 4.4), the domain generator builds the domain of the core presentation problem.

Reactivity is realized via notifications provided by the ACL (see Section 3). If a device leaves the system then its capabilities are no longer available to be part of a presentation. In such a case, the ACL notifies the presentation planner about the device having left and the domain generator removes the respective self-description from the knowledge base. Then the presentation is re-planned with the changed set of devices, beginning from the time point where the event occurred.

The problem generator and the goal state analyzer allow the presentation to appear continuously despite of its adaptivity and its reactivity. The problem generator builds a pair of an initial state and a goal for a single instance of a core presentation problem. Its inputs are the output requests of the dialogue components, the current states of the output components and the current state of the presentation. The goal state analyzer has access to the state of the presentation at any given time. More precisely, the goal state analyzer has access to the states that occur during the solution of the core presentation problem. From such a state, the presentation planner can guess the internal state of the output components. If a new output request occurs or if an event requires changing a running presentation the goal state analyzer gives all necessary information about the current state to the problem generator such that the next presentation fits in seamlessly.

5.4 The AI Planning System

The AI planner has to solve the core presentation problem consisting of causal dependencies, spatial constraints and temporal constraints. In the following we give a short description of the design of the planner in this respect. Note that this section refers to the anticipated final version of our system which differs from demonstrator described in Section 7. There, we state the major differences between both systems.

The solution of a presentation problem requires arithmetic abilities and the specifications of concrete numbers. For example, a presentation has to specify exact coordinates for each presented picture and these coordinates must observe constraints, e.g., the sizes of the available screens and the required minimal resolution to show an image. Such spatial constraints (as well as temporal constraints) are modeled by linear (in)equations. Each action carries equations that have to be satisfied if it is inserted into the presentation and hence its corresponding control schemata are to be executed on a device. A program which finds solutions to a system of linear equations is called a *linear constraint solver*.

Our presentation planner is based on the hierarchical task network planner JSHOP2 [12][17]. As JSHOP2 on its own cannot handle linear constraints, we extended it by the Cassowary linear constraint solver [2] via external function calls. The resulting AI planner considers all constraints within the plan as equally important, i.e., it discards a current intermediate plan as invalid as soon as it discovers a violated constraint, regardless whether this constraint is a spatial, a temporal or a causal one.

It is important for the generation of complex presentations to treat all constraints equally, otherwise some presentation problems cannot be solved. For example, the demonstrator (see Section 7) uses the presentation planner CkuCkuCk [1]. CkuCkuCk uses a two step planning process: The first step uses a HTN planner to generate a presentation plan which solely regards causal relationships. The second step uses Cassowary to solve the linear equations that are contained in the actions of the plan. If these equations are inconsistent then CkuCkuCk fails, i.e., it does not reenter the first step even if there would be a solution with a different plan. Our realization of the AI planner does not have this limitation.

6. RELATED WORK

In the Pebbles project [16] a personal remote controller uses self-descriptions of applications to automatically generate a GUI-speech interface. However apart from the dynamic layout coordination in the GUI, no further output coordination takes place. In the Peach project [15] presentations combining PDAs with a public display are generated. Similar to our system an animated character is used to mediate between both devices. However due to the static museum setting no further IO devices or services can be included into the system.

Braun et al. [5] presented a system which automatically generates multi-device interfaces which are based on a combination of XHTML and XFORMS as well as speech. However apart from the adaptation of the GUI no further output coordination is possible (e.g., synchronizing image effects with speech).

The Easy Living system [7] is used in a home entertainment scenario similar to ours. The system adapts a user interface according to the location and the orientation of the user. The system also tracks the locations of input devices and is able to switch the user interface to another screen. However as a difference to our system it is not possible to split a user interface between two screens. The WWICE infrastructure of the Philips Home Lab [3] allows dynamic interactions between mobile devices and TV sets. But the use of multi-device user interfaces is not supported.

The presentation planner CkuCkuCk [1] is a direct predecessor of our system. It uses HTN planning to find a solution to a presentation problem and uses the linear constraint solver Cassowary to resolve temporal and spatial relationships. Our system advances over CkuCkuCk in two ways: First CkuCkuCk is not able to find solutions to some solvable presentation problems; our system does not have this limitation (cf. Section 5.4). Furthermore, it uses a static approach and does not support adaptivity, reactivity and continuity as our system does.

7. A WORKING EXAMPLE: THE LIVING ROOM SCENARIO

We have implemented a Java-based ACL. It is set on top of the JXTA Java Standard Edition release 2.3.4 [13] and the CyberLink for Java implementations [14] which is the UPnP implementation in Java. Furthermore, we have implemented a demonstrator for the MMO channel which allows validating the presentation concepts.

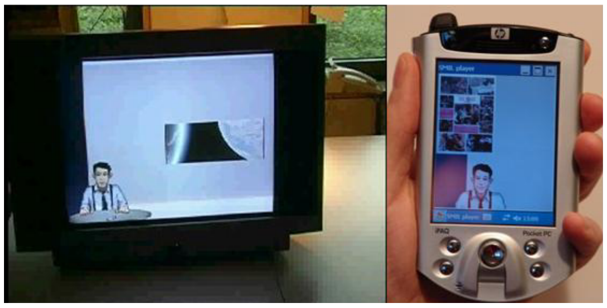

Figure 9: SMIL Presentations involving character and speech on TV set and PDA

Currently the MMO channel demonstrator uses an earlier version of the middleware which does not contain the ACL layer. This means that it is currently not possible for the MMO channel to control devices across UPnP and JXTA networks. Moreover the MMO channel implementation is currently based on the Ckuckuck presentation planner [1] and not the planner described in Section 5. Both integrations will be part of our future work.

The MMO channel was implemented as part of an intelligent living room. The system comprises an HP h5550 iPaq PDA, a DF-1710 17" digital picture frame and a Loewe Aconda TV set with an OnlinePlus add-on. Moreover a Pentium 4 multimedia server is used for computationally expensive tasks involving speech recognition and speech synthesis.

We implemented SMIL output components on the PDA as well as on the TV set. We also implemented an output component for the digital picture frame. This component currently only processes a subset of SMIL which only displays up to one image and one sound file at a time. Additionally we implemented a GUI input component on the PDA. On the multimedia server we implemented a speech recognition input component, a text-to-speech output component and an output component providing the animations of an animated character. More information about these components can be found in [9,10]. We use the self-descriptions described in Section 4.3 to model these components. These self-descriptions are inserted into the planner as part of the presentation context (cf., Figure 8).

We now illustrate the features of the demonstrator. With the system the user can access movie information either by PDA or by speech by saying "Please display movie information". She or he can then choose between several movie titles and then the presentation is displayed on the available output device. The user can choose between a presentation on the TV set and a presentation on the PDA. Both presentations are automatically adapted to the screen resolutions of the devices (Figure 9). If the user switches on the digital picture frame then additional images from the current movie are displayed on its screen (Figure 1). If the PDA and the TV set are switched off then the user is still able to use speech for input and the digital picture frame for output. If the user switches to another TV channel and the SMIL component is not visible anymore then output is redirected to the PDA or to the digital picture frame. If the PDA is available neither then speech output is rerouted to the speakers of the digital picture frame.

8. CONCLUSION

We described our middleware of our intelligent living room scenario which supports ad-hoc spontaneous connectivity between heterogeneous devices connected by different types of networks. We also presented the MMO channel which is responsible for coordinating output components and the MMO presentation planner which uses an AI planning approach to find the best suitable multimedia presentations for the given presentation context. All these concepts are implemented as part of an intelligent living room scenario in which the user can freely choose between different combinations of output devices involving a TV set, a PDA and a digital picture frame.

This approach enables us to control output devices across different types of networks. Moreover the knowledge-based approach of the MMO planner allows specifying presentation strategies which are independent of the resolutions of the output devices and which are able to adapt output content at runtime to the available devices. Finally self-descriptions of output devices enable the MMO channel to integrate new devices in an ad-hoc manner.

In the future we will enhance our demonstrator by new IO devices (for example, by a mobile phone) and services. We will also evaluate the presentation strate/ gies as part of a user study about our demonstrator.

9. ACKNOWLEDGMENTS

The work is funded by the Klaus Tschira Foundation and the German Federal Ministry of Education and Research under the grant 01 ISC 27B for the DynAMITE project. We would like to thank Xiaoming Hu from EML, Michael Hellenschmidt from Fraunhofer-Institute for Computer Graphics, and Jens Neumann from Loewe Opta GmbH for their productive cooperation within DynAMITE.

10. REFERENCES

[1] André E., Baldes S., Kleinbauer T., Rist. T., 2000, *CkuCkuCk 1.01 Planning Multimedia Presentations*, http://www.dfki.de/imedia/miau/software/CkuCkuCk.

[2] Badros G., Borning A., and Stuckey P., 2001, *The Cassowary Linear Arithmetic Constraint Solving Algorithm*. ACM Transactions on Computer-Human Interaction, volume 9, number 4, 267-306.

[3] Baldus, H., Baumeister, M., Eggenhuissen, H., 2000, Montvay, A., Stut, W., *WWICE: An Architecture for In-Home Digital Networks*. Proc. SPIE, Vol. 3969, 196-203.

[4] Bernsen, N., 2001, *Multimodality in Language and Speech systems - from Theory to Design Support Tool*. In Multimodality in Language and Speech Systems. Limerick, Irland.

[5] Braun, E., Hartl, A., Kangasharju, J., and Mühlhäuser, M., 2004, *Single Authoring for Multi-Device Interfaces*, Workshop "User Interfaces For All", Vienna, Austria

[6] Brookshier, D., *Publishing and Discovery of Advertisements,* http://wiki.java.net/bin/view/Jxta/ PublishingDiscoveringPipeAndOtherAdvertiesements.

[7] Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S., 2000, *EasyLiving: Technologies for Intelligent Environments,* Proc. Handheld and Ubiquitous Computing, 12-29.

[8] Elting C. and Möhler G., 2002, *Modeling Output in the EMBASSI Multimodal Dialog System*, Proc. Int. Conf. on Multimodal Interfaces ICMI02, Pittsburgh, PA, 111-116.

[9] Elting, C. and Hellenschmidt, M., 2004, *Strategies for Self-Organization and Multimodal Output Coordination in Distributed Device Environments*, Workshop on AI in Mobile Systems 2004, Nottingham, England, 20-27.

[10] Elting, C., 2005, *Orchestrating Output Devices - Planning Multimedia Presentations for Home Entertainment with Ambient Intelligence*, Smart Objects and Ambient Intelligence sOc-EUSAI, Grenoble, France, 153-158.

[11] Hellenschmidt M., 2005, *Distributed Implementation of a Self-Organizing Appliance Middleware*. Proceedings of the Conference Smart Objects & Ambient Intelligence, Grenoble, France, 201-206.

[12] Ilghami O., 2005, *Documentation for JSHOP2*. Tech-Report CS-TR-4694, Department of Computer Science, University of Maryland, College Park, MD 20742, USA.

[13] The JXTA Project, http://www.jxta.org.

[14] Konno S., 2004, *CyberLink for Java Programming Guide*, http://www.cybergarage.org/net/upnp/java

[15] Kruppa M., 2004, *The Better Remote Control – Multiuser Interaction with Public Displays*, Workshop on Multi-User and Ubiquitous User Interfaces (MU3I), 1-6.

[16] Myers, B. A., Nichols, J., Wobbrock, J. O., and Miller, R. C., 2004, *Taking Handheld Devices to the Next Level*, IEEE Computer, 37(12), 36-43.

[17] Nau D., Tsz-Chiu A., Ilghami O., Kuter U., Murdock W., Wu D., and Yaman F, 2003, *SHOP2: An HTN Planning System.* JAIR, 20, 379-404.

[18] The Universal Plug and Play Forum, http://www.upnp.org.