Extending Universal Plug And Play To Support Self-Organizing Device Ensembles.

Oliver Kutter, Jens Neumann, and Thomas Schmitz

Loewe Opta GmbH
Kompetenzzentrum Hannover
Hildesheimer Str. 140
30173 Hannover, Germany

Abstract. Ad hoc networking technology is being built into consumer electronic devices today and bound to become much more widespread. However, this is but the first step in the strive to bring the ubiquitous computing vision to life.
In this paper we shall take a look at UPnP and assess in how far it should be expanded if it is to be used to move beyond ad hoc networking and towards dynamic, self-organizing device ensembles.
Specifically we shall argue that further differentiation of devices is needed based on the ontologies they belong to. Furthermore it shall be assessed how one might allow the system to determine the best offer if more than one device can perform a given task.

1 Introduction

Electronic, computer-enabled devices abound and proliferate in our everyday lives today. While the underlying technology might be very complex, the handling of these devices must not be so if they are to be useful to a broad public. Furthermore there is a shift from a device-centric perspective that focussed on the capabilities of a single device towards a network-centric view that emphasizes synergetic effects when these devices cooperate.
Existing technology build into consumer devices today focuses on the first step of this shift: Enabling devices to form ad hoc ensembles, provide the user with the option to control some of these devices remotely, and leverage some synergy effect. For example a PDA could be used to control a media server in another room, choose a movie for playback on a TV set in the living room and start the streaming transmission and playback.
This is what Universal Plug and Play (UPnP, [1]) strives to achieve, and mostly does.
The second step then is to aim for a higher degree of automation and enabling a broad array of devices to work together in an ad hoc ensemble in an intelligent fashion. Thus the user interaction becomes more goal-oriented.
For example instead of reaching for the VCR remote, manually inserting an empty tape and pressing the record button, the user could just provide the system with the goal ,,record the current broadcast" (i.e. just press the record

button). The system could then automatically determine the length and other properties of the current program, find out which devices in the ensemble are capable of carrying out the task, automatically choosing the best option and triggering the actual recording.
This second step is part of the focus of the *DynAMITE* research project (Dynamic Adaptive Multi-modal IT Ensembles [2]).

After a short introduction to UPnP we assess in section 3 what is missing if one is to use UPnP for self-organizing device ensembles. Our focus shall be on employing the possibilities the standard offers for our ends, rather than trying to modify UPnP itself. To move beyond ad hoc networking, additional mechanisms are required to enable devices or groups thereof to dynamically coordinate each other in meaningful ways. In our view mainly three points need elaboration: The classification of components to form an ensemble *topology* (sect. 3.1), the introduction of *location* information (sect. 3.2) and a *Utility of Service* (sect. 3.3) to provide means to determine how well an action can be performed by a specific device or service. A summary and an outline of the next steps complete this paper.

2 Universal Plug And Play

UPnP is an emerging standard for ad hoc networking of consumer electronic devices, with broad industry backing from companies such as Microsoft, Intel, Sony, Nokia and several others. The UPnP Forum boasts more than 740 participating vendors ([1]).
UPnP leverages existing technologies, namely TCP, UDP, IP, HTTP, XML and the Simple Object Access Protocol (SOAP) to make it easy for network-enabled devices to offer their services on the network. UPnP differentiates three component types: There are logical *Devices*, which offer *Services* and there are *ControlPoints*. This doesn't have to reflect the physical setup, i.e. a physical device may host several logical UPnP *Devices*. UPnP works in 5 Phases ([3]):

1. **Addressing**. Network devices first need to configure themselves with a valid IP address. This is done via DHCP and, failing that, via AutoIP.
2. **Discovery** is conducted via the Simple Service Discovery Protocol (SSDP). ControlPoints may search for devices and services by type and, in addition, devices and services announce their presence and types regularly.
3. **Description**. SSDP messages contain the location (URIs) of XML formatted device description documents which in turn contain the location of XML service description documents for each service. *ControlPoints* can download these documents using HTTP.
4. **Control**. With the information from the service descriptions *ControlPoints* can control *Services* by sending SOAP messages via HTTP.
5. **Eventing**. Events in UPnP are bound to the Services' state variables. Changes to this state variables are multicasted to all subscribers by a mechanism called *GENA*, which basically consists of HTTP via TCP.

While this works reasonably well for the targeted environment, there are some shortcomings, as listed for example in [4]. Most of these address unnecessary overhead leading to a lot of network traffic or prescriptions limiting design freedom for no good reason.
Despite these deficiencies, UPnP offers, because of its service oriented approach, many advantages over competing technologies, which follow a distributed object approach and typically depend on a specific platform or programming language, usually Java. Independence of hardware platform, operating system and programming language is a must-have for network-technologies designed for heterogeneous device ensembles envisioned in pervasive computing scenarios.

3 Moving Beyond Ad Hoc Networking

3.1 Topology

At the moment all UPnP Working Groups (WG) follow the ,,Everything is a device with services" paradigm. If a functionality, or even just a requirement, is identified or needed, e.g. ,,Light", ,,Video Renderer", ,,Remote Control", ,,Quality of Service" or ,,Security", it is realized as a set of device and/or service specifications.
There is no differentiation between application functionality and infrastructure requirements. For example the ,,Quality of Service" WG defined a number of services that cooperate to provide the necessary QoS networking functionality. In the same way the Audio/Video WG specified devices and services that provide media streaming and rendering.
It is possible to differentiate services by their type, but UPnP provides no additional classification. During Discovery, one might search for a device or service by type or search for all services and devices. It is impossible however to search for all devices or services of some meta class such as ,,output devices".
The ad-hoc network technology could provide mechanisms to differentiate between components of different classes, e.g. application functionalities and infrastructure components.
In our approach we employ the topology published in [5] to realize architectural integration (Fig. 1). This architecture refers to the integration of a device into the communication patterns of the ensemble, whose functionalities can be expressed by means of an ontology. For instance a remote control can be characterized as an input device and automatically attached to an ensemble's *Input event* bus, which makes it possible to route its events to dialog management components.
To support the self-organization of components in a topology as shown in Fig. 1, each participating device is required to offer a ,,channel service", one for each channel it would like to communicate on. A channel in this sense is a message bus for a specific type of message, i.e. Input events, Output Events, etc.
In the above example a *Dialog* component could search and watch out for notifications from input devices implementing the ,,input channel service". Thus it doesn't have to keep track of devices it cannot handle directly. For the exchange of control and event messages on the communication busses we reuse the

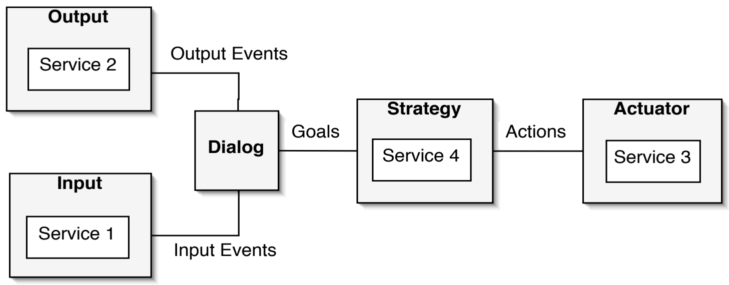

Fig. 1. The DynAMITE Topology differentiates five functional device classes.

UPnP mechanisms. Dialog components control Input-, Output- and Strategy-Components by SOAP Action requests. Strategies in turn control Actuators using SOAP Action requests. In reverse direction the information flows via GENA events.
In comparison to the default UPnP topology which consists just of *ControlPoints* and *Devices*, one can interpret Fig. 1 as partitioning the *ControlPoints* into the four component types Input- , Output, Dialog- and Strategy- Component. Actuators correspond to UPnP Devices, but excluding infrastructure devices and services like QoS and Security. The infrastructure services may be found on any device type.

3.2 Location

In order to make the ubiquitous computing experience as smooth and effortless as possible, some devices will need to be able to announce their location. UPnP does not yet envision this use. A UPnP device or service does not provide its location information, neither in its discovery messages, nor within its descriptions. There is also no ,,UPnP location service" standardized yet. Since location information is not necessarily static and should be evented, it seems advisable to have relevant devices implement such a specialized location service.
Furthermore devices should provide some location info in their discovery messages. Therefore we extend the SSDP message with two additional headers, ,,Mobility" and ,,Locality" which advertise the mobility and locality information of the device, respectively.
This enables a simple way of location based discovery. Exploring the possibilities of this is still work in progress.

3.3 Utility

In future ubiquitous computing environments it will usually be impossible for the user to define or control the behavior of device ensembles, because he would need detailed knowledge of the devices and their abilities.

Moreover device ensembles could be too large and the contexts of use too numerous to cover every situation specifically. It will therefore be necessary for devices to provide meaningful information about themselves and their services in intelligent environments.
UPnP uses its own XML format for description purposes. The information provided is not sufficient for self-organization purposes however.
UPnP service descriptions contain actions with arguments and state variables (light grey classes in 2). In addition to that, information is needed that peers

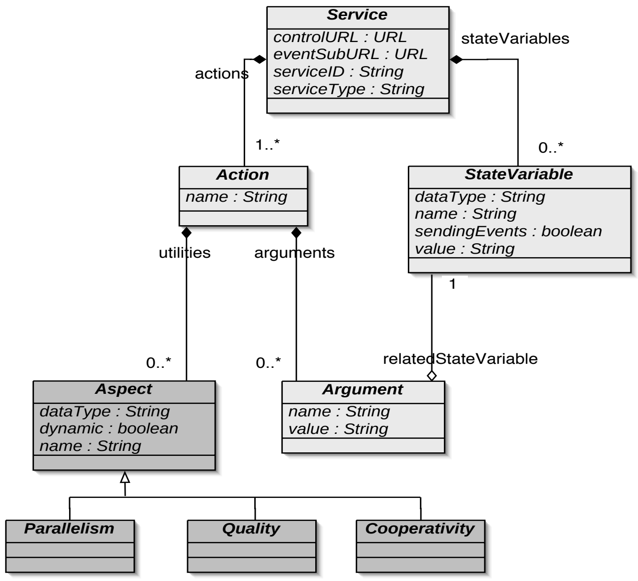

Fig. 2. Aspects for UPnP

looking for a specific type of service may use to arrive at some reasonable assumptions about the available services.
We therefore use extended UPnP service descriptions in DynAMITE which are based on the subscription information worked out in [5]. Apart from the set of actions it is able to execute, a service also provide its utilities (grey classes in Fig. 2), which encode the devices' capabilities to handle each action.
A utility is a set of the following aspects:

- ,,Quality" declares how well the service can process the action
- ,,Parallelism" characterizes to what extend the service can execute the action in parallel to other devices working on the same action
- ,,Cooperativity" specifies whether the services is able to cooperate with other devices in order to process the action

An aspect may be either static or dynamic. If a dynamic aspect is associated with an action, then calling this action does not result in the service processing the action, but in determining and returning its aspect values.
The controlling facilities, i.e. the UPnP *ControlPoints* may then use these aspects to (re-)organize the ensemble or to resolve conflicts. They evaluate the aspects and decide which component(s) will finally process the message.
Values of static aspects are already encoded in the service descriptions and may be evaluated before calling an action.
We are still in the phase of defining the aspects relevant for the example scenarios in DynAMITE. Examples of Qualities, that may be relevant in a ,,Record Movie" scenario are ,,reversibility of recording", ,,medium type", ,,capacity", ,,quality" and ,,noise level of recorder".

4 Conclusion

Ad hoc networking technology such as UPnP does provide decent support for easily using different devices together. It does not foresee strategies to make the device ensemble cooperate in a meaningful way yet.
By leveraging the mechanisms for discovery and description in UPnP, we proposed three approaches to include this information, thereby enabling more sensible device topologies, locality based discovery and providing the system with the means to determine the ,,best offer" for a given service.
Further work will be needed to elaborate on these possibilities and the ,,Best Practices" when using extended UPnP for intelligent, self-organizing device ensembles.

References

1. UPnP Forum http://www.upnp.org
2. The DynAMITE Project http://www.dynamite-project.org
3. UPnP Device Architecture http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf
4. Y. Mazuryk, J.J. Lukkien: ,,Improved Eventing Protocol for Universal Plug and Play" Proceeding of the 5th Progress Symposium on Embedded Systems
5. Michael Hellenschmidt and Thomas Kirste ,,SodaPop: A Software Infrastructure Supporting Self-Organization in Intelligent Environments" Proc. of the 2nd Conference on Industrial Informatics, INDIN 04,Berlin, Germany, 24 - 26. June, 2004.

This work has been partially supported by the German Federal Ministry of Education and Research under the grant signature BMB-F No. FKZ 01 ISC 27A.