Knowledge-Based Middleware for Future Home Networks

Marc-Oliver Pahl
Andreas Müller and Georg Carle
Chair for Network Architectures and Services
Technische Universität München, Germany
{pahl, mueller, carle}@net.in.tum.de

Christoph Niedermeier
Corporate Research and Technologies
Siemens AG, Munich, Germany
christoph.niedermeier@siemens.com

Mario Schuster
Fraunhofer Institute for Open
Communication Systems (FOKUS)
Berlin, Germany
mario.schuster@fokus.fraunhofer.de

***Abstract*—Humans are lazy. They want to get as much support and assistance in their daily life as possible. To provide sophisticated digital butler functionality a system has to monitor the humans and their environment to understand their desires. It has to control all devices to translate the deduced wills into actions. To provide an integrated intelligent environment to the residents it is necessary to overcome the borders of different manufacturers, devices and access technologies. A management and control middleware is needed that shields users as well as high level management services from the technical details of the network of devices by providing a standardized, resilient and secure communication channel.**

We present a knowledge-based middleware as base for that purpose in this paper. The novel approach is the transparent connection of highly heterogeneous home devices by requiring only very limited functionality per device. Our design allows the desired high autonomicity in a secure and efficient way.

I. Introduction

We expect almost every device inside a home (e.g. door bell, fridge, TV, phone, heater) to be connected to the local network in the future. The connection to a common bus allows the components to be monitored and controlled remotely. New forms of collaboration inside a home become possible. Today partial automation for specific domains like air conditioning is in use. Today every manufacturer establishes a proprietary control channel. Thus the different services cannot interact with each other. To support residents in their daily life an organism of acting-together home devices is needed.

The future devices (e.g. appliance, sensor, actor) as well as their access technologies (e.g. Wi-Fi, Bluetooth, ZigBee) will remain highly heterogeneous. The heterogeneity is needed as the different devices inside a future home fulfill various purposes. They differ heavily in their available resources (e.g. computing power, network interface, storage). Therefore, the needs on their control and monitor connection differ.

To make the heterogeneous base of devices manageable for service functionality providers as well as the user, an autonomous control and management platform (ACMP) is needed. It has to provide a secure middleware for the provisioning of smart functionality. The overall goal is to shield the users from the growing technical complexity of their environment. A way to do so is to provide so-called self-X-functionality (see [1]) via the ACMP together with its services.

We present an ACMP that is developed within the German subpart of the AutHoNe[1]-project. Section II outlines the general architecture with a major focus on knowledge representation and sharing. Security aspects of the platform are followed by some use cases. In III we briefly put the approach into the context of three management standards used today. Section IV presents some implementation details of the integration of sensors into the ACMP.

[1]The presented work is part of the AutHoNe project which is partly funded by German Federal Ministry of Education and Research under grant agreement no. 01BN070[2-5]. The project is being carried out as part of the CELTIC initiative within the EUREKA framework. [2]

II. A Novel Autonomous Architecture

Today's networks lack an important functionality we consider essential for future architectures: unified control and management. The proposed architecture adds features to existing networks. It does not replace their functionality. Thus, it is situated between the Data Plane and the Users in Fig.1.

A. Architecture Model

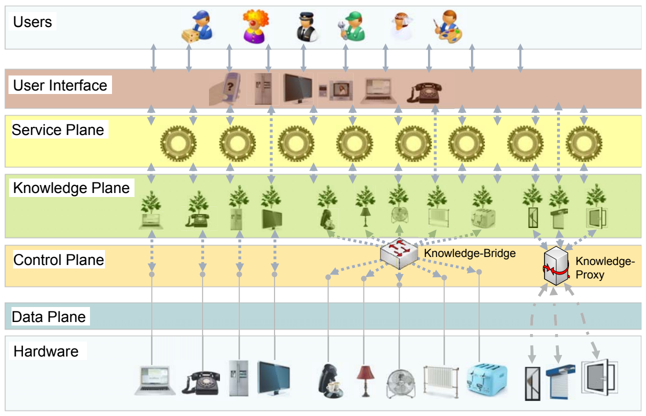

Fig. 1. Functional planes of the proposed architecture

The proposed control and management platform consists of three logical planes (see Fig.1):

- Service Plane
- Knowledge Plane
- Control Plane

The platform provides abstraction over the hardware and allows to share functionality between services in an easy way. We start our presentation with the most abstract of the functional layers. Then we go layer by layer down to the hardware.

1) Service Plane: The service plane is responsible for high level control and management tasks. Therefore, it is situated directly under the *User Interface* that is interacting with the user. It orchestrates the home devices by interpreting the user's will and transforming it into actions. The service plane gets the input parameters for its inference and decision processes from the central layer: the knowledge plane.

The interface to the knowledge plane is *unified* for all participating entities. This is an important concept as it allows to plug together everything that is representable inside the knowledge plane. Via the plugging, sophisticated functionality can be provided without having to care about tasks that are already handled by other services.

The services on the service plane are decoupled from a node's hardware by the abstraction layer knowledge plane. This makes them migratable from one node to another.

A typical task of a service is the control of a room's temperature (see Fig. 2). To perform that task the control service acquires temperature information of the room through the knowledge plane. The service then runs some logic and informs an appropriate node like the heater about necessary steps to undertake via the knowledge plane.

As we will see in the description of the control plane, the heater decides autonomously what to do with the information "it should get warmer". This results in a loosely coupled network of distributed intelligent nodes.

2) Knowledge Plane: The knowledge plane is the central layer of the architecture. It is responsible for brokering information between nodes allowing the distributed use of data, only available locally today. Establishing a central knowledge layer, which is mandatory to communicate with hardware devices as well as services, makes it possible to distribute functionality over the whole network. This is a key feature for automating domains that do not have enough computational power to perform sophisticated control and monitoring tasks. It enables software using the platform to take all available input data into account to support the resident in every way that is technically possible. In the future, when the usability gets facilitated through automation, not only computer scientists will live in electronic homes, but everyone. The knowledge plane is the backbone for knowledge-based distributed autonomic management (see Sec. II-A3).

The representation of the control and manageable objects in the ACMP is called model. It is important that the semantics of the models are understandable to all attached nodes. Black box knowledge has only little use as it is not usable for most of the services.

The knowledge inside the models is organized in trees. Each node connected to the ACMP exposes its so-called *node tree* (see pictograms in Fig.1). The ACMP-interaction happens only through the object's model that is described by the node tree. Models reside logically inside the knowledge plane.

Services inside the service plane only communicate with each other and the hardware nodes through models. Services for general tasks, like the control of heaters, can only be built if the semantics of leaves inside the node trees are non vendor specific: the services must be able to "understand" the available information.

Therefore, a *hierarchical order of several trees* is proposed. Inside a global register *general trees* for specific devices (e.g. specific type of notebook) are provided. The general trees are formed out of standard elements like a switch, a port, a screen, etc. Using existing standard elements as building blocks, to represent a specific device, leads to a unification of the so-called general tree entities.

In the general trees we have models for all hard- and software, controllable by services inside the ACMP. Since the general trees are available globally they can be used by service authors to identify which kind of leafs inside a home network they want to interact with – prior to the provisioning in a specific home. The semantics of the trees are unified and vendor comprehensive, as well as available to service authors. In a home, network node trees are an instantiation and aggregation of specific entities represented in a general tree. All trees inside the knowledge plane are described within a common format.

To be useful, knowledge has to be discoverable. The discovery mechanism ("is a specific knowledge item available") is a frequent operation. To make the discovery resilient and fast several caching mechanisms are provided for normal nodes as well as replication mechanisms in special nodes.

Discovery is done via multicast. Groups are formed by function (e.g. group heads that contain structure information of their subjects) or topic (e.g. all alarm systems). In combination with a flexible publish-subscribe mechanism this allows the specification of knowledge filters, ensuring that only knowledge that is actually required is being delivered to a particular node. Grouping is one mechanism to control the dissemination of knowledge.

Knowledge is stored in so-called *Knowledge Stores*. When possible it is saved where it occurs (principle of locality). Caching on knowledge stores located on special nodes inside the ACMP is intended. Additionally to replication (resilience, speedup) these external stores can provide more data than a (possibly space limited) node (e.g. the history of the room temperature over the last year).

The connector between a node and the knowledge plane is called *Knowledge Agent* (KA). It publishes knowledge items on behalf of local publishers, and forwards knowledge items received from local publishers or publishers on other nodes to local subscribers (e.g. services).

The KAs are connected via the *Knowledge Bus* that spans the knowledge plane. The knowledge bus uses the node's existing network connection as underlying infrastructure. Services on a node may contribute to the local and distributed knowledge base by producing knowledge and publishing it locally or via the knowledge bus.

The abstraction over the node's model makes it possible to manage a node with a very limited set of commands: get, set, (un-) subscribe and notify (with value). This interface is provided by the KA to local services as well as to remote KAs. The uniformity and simplicity of the KA-interface is very important to keep the complex underlying network of devices manageable. Especially to provide a comprehensible-secure communication channel, the architecture must be simple. Finally, the interface to interact with the knowledge plane must be as easy to understand and use as possible for service providers and hardware vendors since they will not use the ACMP otherwise.

Having the KA as single communication point between the node and the knowledge plane allows the execution of tasks common to all services running on a node like the enforcement of security policies or caching of data.

The KA makes the knowledge plane access to models of other services and hardware transparent to the services running on a node. This abstraction allows them to run on every node inside the network or even to be moved.

The described mechanisms allow the coupling of autonomic devices and services. A distributed network of knowledge-based services emerges allowing tasks to be handled on different nodes collaboratively. The publish-subscribe approach allows synchronizing the loosely coupled entities.

As we have seen how the high level functionality inside the services interacts with the models inside the knowledge plane and how knowledge is organized inside the knowledge plane, we will look at the connection of the devices to be control

3) Control Plane: The control plane integrates the distributed system of autonomic hardware nodes into the universal knowledge sharing infrastructure. Each round dot inside the control plane in Fig.1 represents a node's KA with its underlaying autonomic manager. Fig. 2 shows possible components of an autonomic manager.

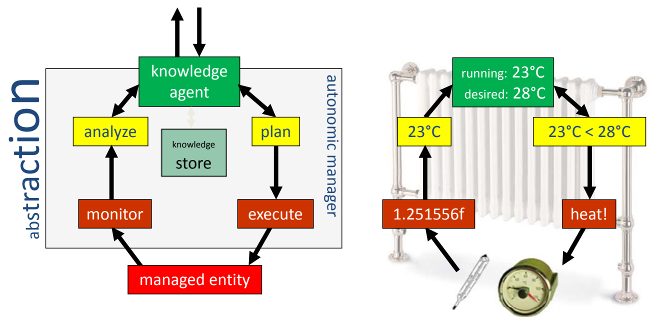

Fig. 2. MAPE-cycle integrated with knowledge sharing mechanism

The left part of Fig. 2 shows the structure of an autonomic manager that is integrated with a KA. The structure depicted in the diagram has been inspired by the MAPE (Monitor, Analyze, Plan, Execute) model proposed by IBM [1]. However, rather than designing it as a closed cycle, where information is directly fed from the analysis into the planning module, the KA has been inserted connecting the autonomic manager(s) to the knowledge plane. The responsibilities of all entities not already introduced will be described below. The right part of Fig. 2 shows a simple example how the proposed architecture may be applied for a concrete autonomic control task.

A *managed entity* is a physical device or a software module that is monitored and/or controlled by an autonomic manager. For that purpose, the managed entity must exhibit so-called *sensors* and *actuators* (not necessarily in a physical sense). In Fig. 2, a physical sensor (a thermometer) and a physical actuator (a valve) are used to control a physical managed entity (a heater).

A *monitor* module is responsible for periodic reading and processing of raw sensor data. It feeds the processed data into an associated *analyzer* module that performs analysis and processing and occasionally publishes the result of the processing as knowledge to the KA. The analyzer provides the knowledge for the model of the managed object to the knowledge plane. In the example, the monitor measures the temperature and converts the raw data into reasonable units. The analyzer verifies the temperature, occasionally fuses it with measurements from other sensors and publishes the result as actual room temperature to his KA.

A *planner* is responsible for receiving knowledge events and performing decision making steps that occasionally lead to actions performed by an *executor*. The planner is triggered by a knowledge event and may perform his decision making based on further knowledge requested/obtained from the KA. The executor receives instructions from the planner and performs according actions on managed entities associated to it. It is important to note that there is no direct connection between the KA and the executor. Changes in the model are not directly reflected to the controlled object. The planner is deciding autonomously if and when it wants to perform desired steps. This is a notable difference to existing control and management frameworks as it enables the distribution of decisions and the independence of components from a central control station. A planner might for example get the order to change the IP address of a component. After having decided to do so, the planner might check if connectivity to the network is still given. If not it will switch back to the last running configuration reconnecting the device to the knowledge platform. Without this autonomic technology it would have been necessary to reintegrate the device manually back into the platform. Similar examples where autonomicity is a clear advantage can be thought of. In Fig. 2, a planner receives the desired room temperature and the actual room temperature from the knowledge agent, performs a comparison and instructs the executor to handle the valve of the heater if necessary.

Each node is equipped with one or several *autonomic managers* that are responsible for node-local monitoring and analysis as well as planning and execution activities. Analysis and planning activities may take into account knowledge that has been acquired locally or has been collected via the knowledge bus. It is important to note that the two mechanisms are intended to be only loosely coupled. That means that at each point of time, autonomic management is only relying on information that is available locally; no communication with other nodes is required for performing local analysis or decision making steps. On the other hand, knowledge sharing is usually triggered by time (periodically) or an (asynchronous) event according to subscriptions of particular types of knowledge. These subscriptions are geared to long-term tasks rather than short-term information needs.

Nevertheless, these loosely coupled mechanisms are suitable for realizing collaborative decision making processes. This is achieved by considering certain knowledge items as events that are notified to their respective subscribers. Any module of the node software that provides sufficiently sophisticated processing logic may act as a subscriber to knowledge events. Examples are components of autonomic managers as analysis or planning modules. A subscriber may react to such a knowledge event by performing an analysis activity and publishing the result as another knowledge event via the knowledge bus. All subscribers of that result may in turn react on the latter event and perform either further analysis or a planning activity leading to an execution step. Thus, collaborative distributed analysis and planning activities may be triggered by coupling node-local processing steps using distributed knowledge events.

Having described the functionality of the autonomic manager on an ACMP node, we introduce two more concepts that are necessary to connect a broad base of devices to the platform: the knowledge bridge and the knowledge proxy (see Fig. 1). Some implementation details of the described concepts will be illustrated in Sec. IV.

The *Knowledge Bridge* translates between fully functional nodes inside our ACMP, running a KA and a KS, and reduced functional nodes, running a possibly minimal KA. The reduced functional nodes typically have low resources (computing power, energy, network connectivity and storage) allowing them to run a limited set of ACMP functionality only. Often this implies that only an adapted version of the knowledge exchange protocol is used in such a domain as we will see in Sec. IV. The class of reduced functional nodes is important in the home context since it represents a majority of devices. Nodes consisting of a sensor, an actor and a computationally weak processor are not able to provide the full functionality that is needed to be directly connected to the knowledge platform. They can provide limited functionality. The knowledge bridge is used to make such nodes transparently controllable from inside the fully functional knowledge platform. It transcodes knowledge on its way between fully functional nodes and reduced functional nodes transparently. This allows the temperature control to run on a powerful node outside the sensor domain for instance. It enables new functionality for domains lacking of needed resources for a feature.

Besides the nodes that run partial ACMP functionality we will always have legacy nodes that are not running any of our platform specific functionality. If these devices provide control and management functionality in a proprietary way they should be integrated into the network. This happens through the devices called *Knowledge Proxy*. While the knowledge bridge only changes the information that passes it, the Knowledge Proxy provides ACMP-functionality for the devices connected. An example for such legacy nodes are nodes running a SNMP-agent. Other examples are the before mentioned proprietary controls for air conditioning or heating. Autonomicity is an important concept for ACMP nodes. The knowledge proxy therefore has to provide autnonomic funtionality behalf of the connected device if not already existing there. In case of SNMP no autonomic functionality is available on a normal SNMP agent. The proxy will run an autonomic manager for every SNMP device connected to it. All legacy devices are integrated in a transparent way by that concept. This is important to get a broad base of controllable and manageable devices. Only with that base extensive digital assistance becomes possible and a critical mass for the introduction of a system as the proposed one can be reached.

B. *Security*

Control and management tasks inside a home network are highly security-critical since many of the controlled entities are situated in the real world. The actions are not only influencing software components but especially security-related hardware like the opener of the front door.

Securing the ACMP provided functionality is essential to make it applicable for network management. The ACMP has to make sure that the *correct commands are sent to the correct device by the correct sender*. To reach that required goal the *data flow* on the communication channel has to be *secured*, the *authentication* of the communication partners has to be proved and *authorization* mechanisms have to be provided by the platform. As stated earlier the design of the ACMP is kept as simple as possible to be able to provide comprehensible security. This section focuses on a solution for providing provable identity to devices. The proved identity is then used for authorization.

A two level hierarchy of cryptographic identifiers is introduced. The higher level is the homeID that identifies each home context. To get a globally unique identifier it gets concatenated with the site unique deviceID.

For authentication as well as encryption reasons each device and each home gets a public/private key pair. The IDs are derived from this public key.

The KAs, as the entities exchanging information that should be verified, automatically prove their identity with a data exchange involving the private key. To prove the belonging of a certain KA to the home context, each new device gets a certificate along with its public private key pair. Thereby the identity gets related to a common trust anchor inside the home. For that purpose one node inside each home network becomes a Certificate Authority (CA) for its devices. When a new device/node gets registered inside a home a certificate is issued automatically for it. The ownership of a certificate makes it unnecessary to query a register each time the verification of a device belonging to a home has to be proved.

The platform (KA) performs the described identity check automatically, the authenticity of the communication partners is assured. Receiving the ID as well as the certificate of a node, any KA that possesses the home's public key can verify the membership of the node to the home. This knowledge can be used to authorize the communication partner to access parts of its knowledge.

The homeID is globally unique so each device can be unambiguously identified through its concatenated identifier. The use of the homeID as part of the address relates the device to its home CA. This enables the use of devices in foreign home networks. Nodes inside a foreign ACMP instance can perform the same verification steps local nodes do.

A foreign node has to know the public key of the visiting node's home to validate the node certificate. For that purpose, an exchange mechanism similar to zfone [3] was implemented. Two mobile devices can exchange their home's public keys when meeting. The exchange is secured by an additional one time secret that has to be communicated over a secure second channel (e.g. a PIN exchanged via voice).

For enabling revocation of device certificates (e.g. stolen ID, sold device) the initial authentication of a node inside a foreign ACMP instance can be mandatory combined with contacting the CA in the node's home. Trusted Computing [4] technology (Trusted Platform Modules) is used to prevent the theft or manipulation of keys.

C. Use Cases

The application areas for the presented platform are manifold. It can be used as a solid base for the provisioning of services that assist us in our daily life as it provides a secure unified and scalable knowledge exchange mechanism fitting for the different domains with their different resources we find inside a home. There are currently several fields that are promising for home automation:

One is the *building security*. As the platform provides appropriate security mechanisms it can be used to control the active alarm systems like shutters or lights as well as monitor the passive ones like emergency sensors. The integration of more devices (also non-special alarm system sensors) allows new functionality by strengthening the home security.

A second field is the *electronic health*. The presented platform can become a valuable part of an electronic health system not only by monitoring a humans environment (or even the human itself) but also by controlling the environment and thereby making it more healthy for the resident (e.g. by adding fresh air).

A third important field is *energy saving*. Again the unified control and management functionality over domain borders comes into play. The possibility to remotely control the energy consumers allows the deployment of energy services that are effective. A service that ensures that all devices that are not needed are off, when no one is in the house, will save a lot of energy worldwide. In combination with cooling and heating systems, the amount of saved energy will be even higher.

A fourth aspect is *comfort*. In combination with artificial intelligence that takes the monitored data as base, the platform can be used to provide a level of comfort to the user that is not possible today as it requires for instance special engineering knowledge just to connect the different domains of controllable devices.

As stated the concept of the platform enables developers to build sophisticated services without having to handle common aspects from scratch. This will not only lead to new software services, but also to new hardware (as the usability through automatic integration significantly improves), thus raising the acceptance as well. With real plug and play, even technophobic people will be more willing to invest in new technical equipment for their home.

Many other application areas are affected as the provided functionality just lays the solid base and does not limit the creativity of developers.

III. Related work

A vast amount of information-brokering middleware and agent platforms exists. Our platform has a different approach from many of them: it provides only the necessary minimum of functionality. Features already integrated in other middlewares are added as modules, providing clear interfaces and flexibility. Our work leads into the direction of a management standard for autonomous control and management. Therefore, we will not compare our platform to existing agent platforms here but have a brief look at control and management software used in today's networks.

Today's most widely used management framework is SNMP [5]. Even though it provides a data model that is similar to our trees, it provides no autonomous and decentralized functionality. It does not support autonomous management.

The WBEM [6] architecture with WMI as one of its implementations is another current management framework. Again autonomous functionality is not addressed. Additionally, the complexity of the framework is too high to run on the reduced functional devices in our domain.

NETCONF [7] in combination with YANG [8] as datamodel is the third management solution that experiences some acceptance. It is currently predominantly used for management of routers. Even though its domain is very different from ours we could implement a prototype of our ACMP staying almost conform with the (proposed) standards.

As we can see the three major control and management tools used today do not fit the domain addressed in this paper. They do not fit for autonomicity in general as well. Therefore, we see the need for a new standard for autonomous control and management.

IV. An ACMP Proxy towards Sensors

The Wireless Sensor and Actuator Network domain is very important in the home context since many of its devices are sensors. In this section we have a short look at the implementation of a knowledge proxy that extends the ACMP towards sensors.

On the top left of Fig. 3 we see some general trees. Those on the bottom right are for sensors. Concepts of the Sensor Model Language (SensorML) [9], Transducer Markup Language (TransducerML) [10] as well as IEEE 1451 [11] were used to build appropriate model structures for sensors.

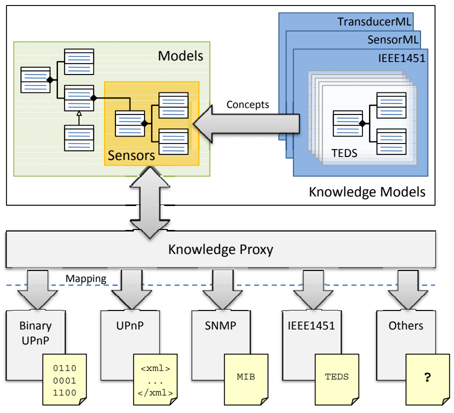

Fig. 3. Knowledge bridge implementation

On the bottom we see different protocols the proxy uses to communicate with sensors. As stated before it is necessary to transform the information from the device into the representation of the ACMP and vice-versa. To better understand this we have a closer look at an SNMP proxy in Fig. 4.

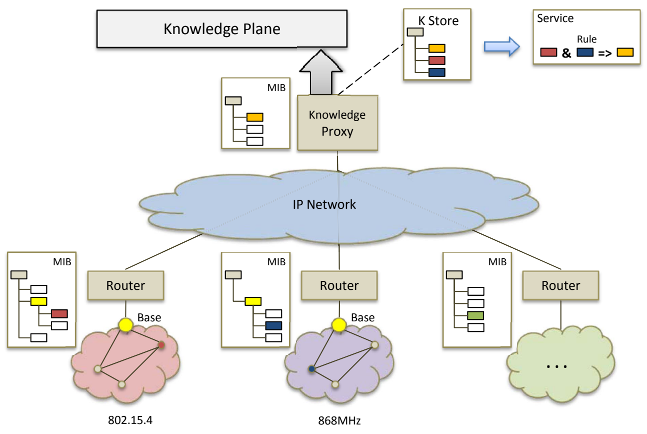

Fig. 4. Storing shared tree information in SNMP-MIBs

As we can see the information of the different nodes (bottom) is transferred over the SNMP agents running on the devices to the proxy. As depicted on he right, the proxy is running some control logic. This is an example where a machine with more resources performs control tasks for weaker devices. As mentioned before the control logic could be distributed all over the network as it is logically situated on top of the knowledge plane.

V. Conclusion and Further Work

The presented autonomic control and management platform is adopted to the specific needs of the home domain. It handles all kinds of controllable devices and provides appropriate security. It provides real autonomic management as the nodes are fully decoupled from eachother. This distinguishes it clearly from existing management frameworks.

As stated at the beginning we expect the sector of home networking to grow rapidly over the next few years. The availability of cheap communication modules will lead to a broad base of (installed) devices that are remote controllable. To benefit from the emerging possibilities the devices have to be able to interact vendor comprehensive. For that purpose a solid standard is needed. As soon as an integrated solution for the control and management of all devices exists, the users will see the advantages of the integrated approach over the service isles we have today and take it into account for their buying decision. As (pluggable) standard functionality comes already out of the intelligent network prices may lower.

The success of a platform, as we presented it in this paper, does not come out of the platform itself. It comes from the devices and services available for it. Our platform is ready to integrate all devices, if they are running our software or not. We expect commercial software authors as well as free programmers to provide drivers (for new and existing devices) and services. Our simple interfaces and concepts that can be understood easily in short time are a promising prerequisite for that. We envision a market place on the net where services as well as hardware drivers for the platform are provided. Getting gadgets for the home will be as easy as loading widgets for the iPhone today.

In parallel to the platform we are currently developping services to provide autonomous functionality on top.

Things that are available today for technical experts only become available to everyone in the future.

References

[1] Jeffrey O. Kephart, David M. Chess, *The Vision of Autonomic Computing* Computer, vol. 36, no. 1, pp. 41-50, January, 2003
[2] AutHoNe-DE Consortium, *AutHoNe-DE Project - Home Page*, WWW, 2009, http://www.authone.de/
[3] P. Zimmermann, A. Johnston, et al., *ZRTP: Media Path Key Agreement for Secure RTP*, IETF Internet Draft (work in progress), draft-zimmermann-avt-zrtp-15, 2009
[4] The Trusted Computing Group, *Trusted Computing Group - Homepage*, WWW, 2009, http://www.trustedcomputinggroup.org/
[5] J. Case, M. Fedor, M. Schoffstall, J. Davin, *A Simple Network Management Protocol (SNMP)*, Request for Comments (RFC): 1157, IETF, 1990
[6] Distributed Management Task Force, *Web-Based Enterprise Management (WBEM)*, 2009 http://www.dmtf.org
[7] R. Enns and M. Bjorklund and J. Schoenwaelder and A. Bierman, *NETCONF Configuration Protocol*, Internet-Draft 4741bis-01, IETF, 2009
[8] M. Bjorklund, *YANG - A data modeling language for NETCONF*, 2009, http://www.yang-central.org/
[9] *Introduction to SensorML*, WWW, 2009, http://vast.nsstc.uah.edu/SensorML/
[10] IRIS Group - Innovative Research Ideas and Services, *Transducer Markup Language - Homepage*, WWW, 2009, http://www.transducerml.org/
[11] *NIST IEEE-P1451 Draft Standard Home Page*, WWW, 2009, http://ieee1451.nist.gov/