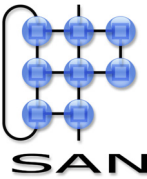


A Service Oriented Architecture for Ambient Intelligence Choreography and Secure Service Discovery

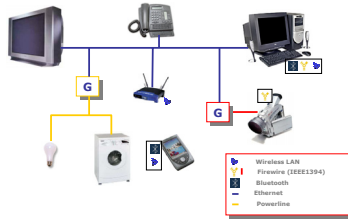
dr. J.J. Lukkien, ir. R.P. Bosman, M.Tjong PDEng, dr. ir. P.H.F.M. Verhoeven

Internet-enabled Monitoring and Control of Embedded Systems (PROGRESS project EES.5413)



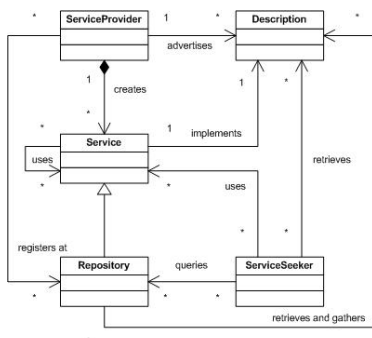
1. Context: Ambient Intelligence

Ambient Intelligence (AmI) represents a vision of ubiquitous computing, sensing and actuating to unobtrusively enhance everyday life. AmI relies on cooperation of devices not foreseen at design time. We investigate an open software architecture that separates functionality from coordination, within the context of heterogeneous devices and heterogeneous connections.



2. Basis: Service Oriented Architecture

Service Oriented Architecture (SOA) is an architectural style that envisions functionality as networked services. In a SOA, typically service providers publish services at a service repository and service consumers look for services by querying repositories. The services are thus discoverable.



Moreover, services in a SOA are *loosely coupled* and *bound at a late stage* allowing runtime replacements of services. These properties make SOA a good candidate to realize the AmI vision.

Two challenges we consider in realizing the AmI vision with SOA are:

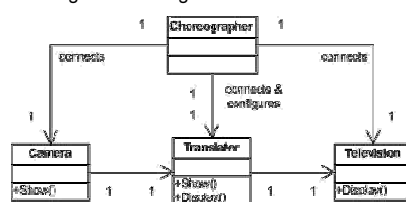
- to express applications in AmI domain in terms of cooperating services
- to deal with privacy and security issues in a ubiquitous environment

3. Choreographed applications

Applications in a SOA (environment) are usually defined by composition (created by executable orchestrators) or coordination (specified by choreographies) of services.

Choreographies are typically implemented manually by all involved parties. Instead, we define executable choreographers which establish a (service) choreography by performing the following tasks:

- discovery
- binding
- configuration
- monitoring



The established applications can change dynamically based on context and network dynamics.

Affiliation:

Eindhoven University of Technology / Department of Mathematics and Computer Science
 SAN Group, P.O. Box 513, NL-5600 MB, Eindhoven The Netherlands
 +31 40 247 8309 / +31 40 247 8345 / wsinsan@tue.nl / http://www.win.tue.nl/san/

In order to support flexible bindings we consider three areas of service protocol incompatibility:

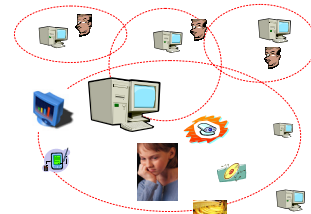
- message format (bridging standards like UPnP, web services, CORBA)
- interface compatibility (mapping method names and parameter lists)
- interaction compatibility (message order, communication patterns)

The choreographer will tackle above compatibilities by deploying translators (configurable orchestrators) to bridge these gaps.

Challenges: choreographer description and execution, resolving service incompatibilities.

4. Virtual communities for secure service discovery

To provide security and privacy, we consider using virtual communities as the scope of service discovery process. A virtual community comprises a number of people that have a commonality and share their services with each other.

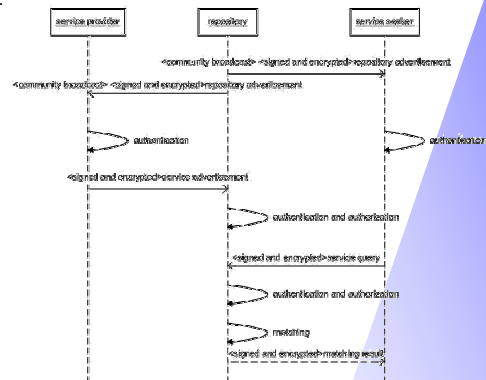


Service discovery within virtual communities supports the following properties:

- Privacy: controlling who is allowed to know about one's services
- Security: authorized and authenticated service access, non-repudiation, confidentiality and integrity.

Privacy and security is achieved by supplying community members with credentials that will limit the service discovery and usage to authorized members only.

Challenges: allowing overlapping communities and different service access levels.



5. Current and future work

Open service architecture providing:

- secure service discovery
- application definition
- context awareness

References:

1. Universal Plug and Play: www.upnp.org
2. Web services: www.w3c.org
3. Y. Mazuryk, J.J. Lukkien, *Analysis and improvements of the eventing protocol for universal plug and play*, Proceedings of the 2004 international conference on Internet, communication and information technology, St. Thomas, Virgin Islands; November 2004