SAASHA: a Self-Adaptable Agent System for Home Automation

Fady HAMOUI*, Marianne HUCHARD+, Christelle URTADO* and Sylvain VAUTTIER*

Home Automation Systems

Smart homes are equipped with networked domestic electronic **devices** that each provide several **services** and are remotely controlled by a **home automation system**.

State-of-the-art home automation systems often are capable of playing **simple** and mostly **predefined scenarios** that sequentially execute various devices' services and are useful in recurrent situations.

They generally fail to:

- ❖ provide **inexperienced users** with the ability to define their own **rich scenarios** and have them **automatically and dynamically implemented** so they can instantly be played,
- ❖ **perceive their environment**, and **changes** in their environment, and **adapt to them**.

Requirements for SAASHA

Expected qualities of the proposed system thus are:

- ❖ configurability,
- ❖ context-awareness,
- ❖ autonomic reconfigurability,
- ❖ dynamic adaptability.

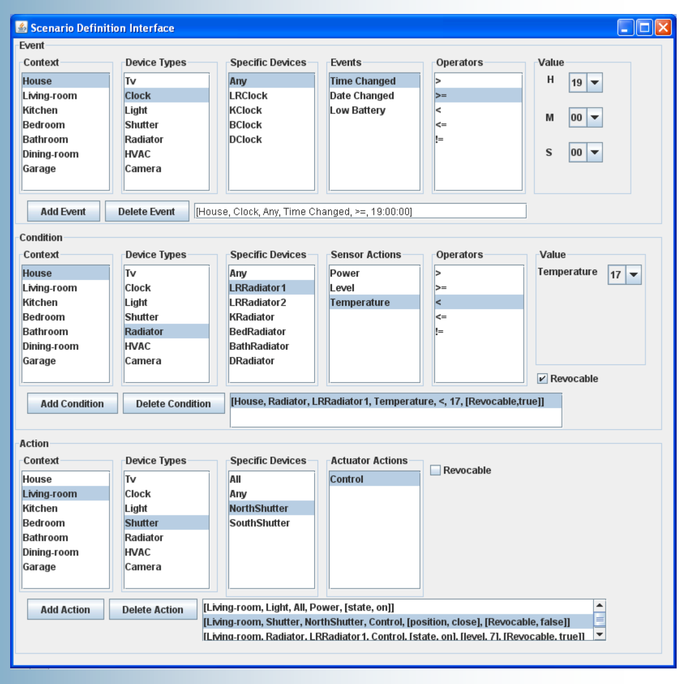
Sample scenario. Close all shutters, turn lights on in the living-room and turn radiators on as soon as night falls if house temperature is too low.

Architecture of SAASHA

SAASHA is **agent-based.**

- ❖ **Graphical User Interface Agents** mediate the interaction with users. Administrators can **parameterize** the system. End-users can **define custom scenarios.**
- ❖ **Device Control Agents** control devices' **service execution**, and **implement scenarios**.

Rich scenarios are defined as **ECA rules.** As soon as some **Event** occurs, if **Conditions** all are satisfied, the associated sequence of **Actions** is executed.



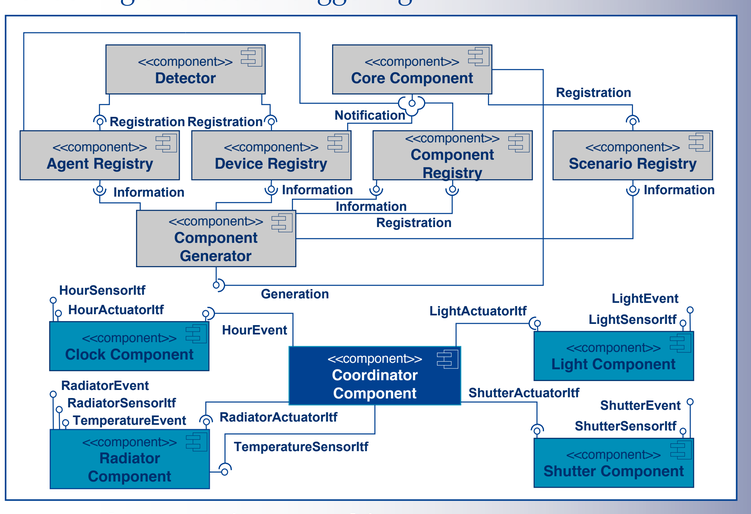
Scenario definition GUI.
*Users select events, conditions and actions that are **automatically extracted** and **classified** from device and service descriptors in a **dynamically updated** GUI window.*

SAASHA agents are made from **software components.**

- ❖ **Meta-level components** are the common part of all agents' inner architecture. They implement **agents' common behaviors**.
- ❖ **Control components** are **dynamically generated** from device and service descriptors to **wrap device drivers**.

They are **automatically deployed** into their destination agents' architecture and **control the devices** their destination agents are responsible of.

- ❖ **Coordination components** are **dynamically generated** from scenario descriptions. They are **automatically deployed** depending on both control responsibility repartition and administrator-parameterized **deployment strategy**. Alone (in case of centralized scenario deployment) or in groups (in case of distributed scenario deployment), coordination components **implement scenarios**. They are responsible of event occurrence monitoring, condition checking and action triggering.



A Device Control Agent's inner architecture.
Meta-level components (grey) are the agent's predefined part. Turquoise blue components are device control components. They are added into the agent's architecture to support a specific scenario that involves a shutter, a light, a radiator and a clock. The coordinator component (deep blue) is deployed to coordinate the behavior of all components involved in the scenario. This example scenario implementation is centralized.

SAASHA provides means to **avoid scenario conflicts**.

SAASHA **reconfigures itself** (finds alternate possibilities, re-generates components and re-deploys them) so as to try and **maintain service continuity** when agents or devices fail or become unavailable.

Implementation of SAASHA

SAASHA's implementation combines the use of **UPnP** as a middleware for **device and service discovery and control** and of **OSGi** as a framework for **component dynamic deployment.** SAASHA agents are themselves coded as UPnP devices.

The **Domus simulator** demonstrates the effects of SAASHA scenario execution on virtual devices when real-world devices are not available.

Snapshot of the Domus smart home simulator.
Domus is a JavaFX GUI that provides a realistic view of a house and of installed UPnP virtual devices (here, the lamp, air con, washing machine and shutters) and enables to act on devices through their views.

Conclusion

SAASHA **automatically and dynamically adapts itself to its surrounding** physical environment **without previous knowledge**.

SAASHA **detects, identifies and controls available devices**. It provides inexperienced users with an adapted **scenario definition** GUI. It **implements, deploys and executes user-defined scenarios**.

Administrators can optionally **parameterize** the running mode of SAASHA.

These capabilities are enabled by the **automatic generation and dynamic deployment of device control and coordination software components**.

Euromicro SEAA 2010, Lille, France

*LGI2P, Ecole des Mines d'Alès, Nîmes, France
{Fady.Hamoui, Christelle.Urtado, Sylvain.Vauttier}@mines-ales.fr

+ LIRMM, UMR 5506, CNRS and UM2, Montpellier, France
huchard@lirmm.fr