

# Distributed Smart 2pace Orchestration System

Marc-Oliver Pahl | pahl@net.in.tum.de | Technische Universität München | Network Architectures and Services | Prof. Dr.-Ing. G. Carle

## Structure and Facilitate Smart Space Orchestration

### User

Users do not only have the role of *classical users* in DS2OS. The system is designed to enable *participation in the development* of components.



develop & share

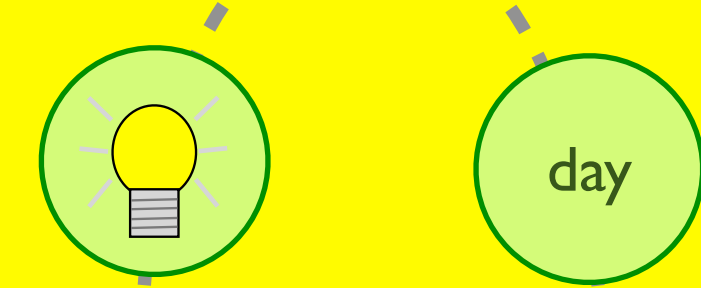
use

### Services

Services are all software components that can be plugged to the VSL. DS2OS realises a service oriented architecture (SOA) with the VSL tuple space for inter-service communication. DS2OS services can be categorised into *orchestration services*, containing orchestration logic, *advanced reasoning services*, containing functionality to extend information in the VSL, *user interface services* that interact with users, and other services.

#### Advanced Reasoning

Advanced Reasoning services reason about existing information to create new information. E.g. data processing in combination with a GW.



Connector

#### Orchestration

```
newMotionSampleCB:
  if (get("/.../isMotion")==1)
    d_lastMotionDetected = 0;
  else
    d_lastMotionDetected += d_lastCallBack;
    if (d_lastMotionDetected > d_preset)
      set("/.../lamp23/isOn", 0);
main:
  register;
  d_lastMotionDetected = 0;
  subscribe("/.../isMotion", newMotionSampleCB);
```

Easy to understand, well structured code

Connector

#### User Interface



Connector

### App Store

The introduction of models as abstraction on top of specific devices allows the creation of *portable services* that can be shared. The App Store is a global repository for smart space services similar to the App Store for smartphones. Different smart spaces are very heterogeneous in their devices. Therefore each service in the repository contains a *description of models* that must be available in a space to run the service.

### Virtual State Layer (VSL)

The Virtual State Layer (VSL) is the core of DS2OS. It is a *distributed tuple space* combined with a *publish-subscribe* system that notifies subscribing services on changes of a tuple. The tuples inside the VSL are *hierarchically structured*. The resulting sub trees are typed. Types are defined in the Model Repository. The VSL is provided by Knowledge Agents (KA). They offer the DS2OS programming interface (API) to service developers. The API has a *fixed set of methods*. The VSL offers *persistence* to all services using it. It takes care of the *access control* and the *communication* between the distributed entities of the system. These aspects *facilitate developing software* for DS2OS. KAs run on each machine in the network that will host services.

#### Knowledge Agent

API with fixed methods

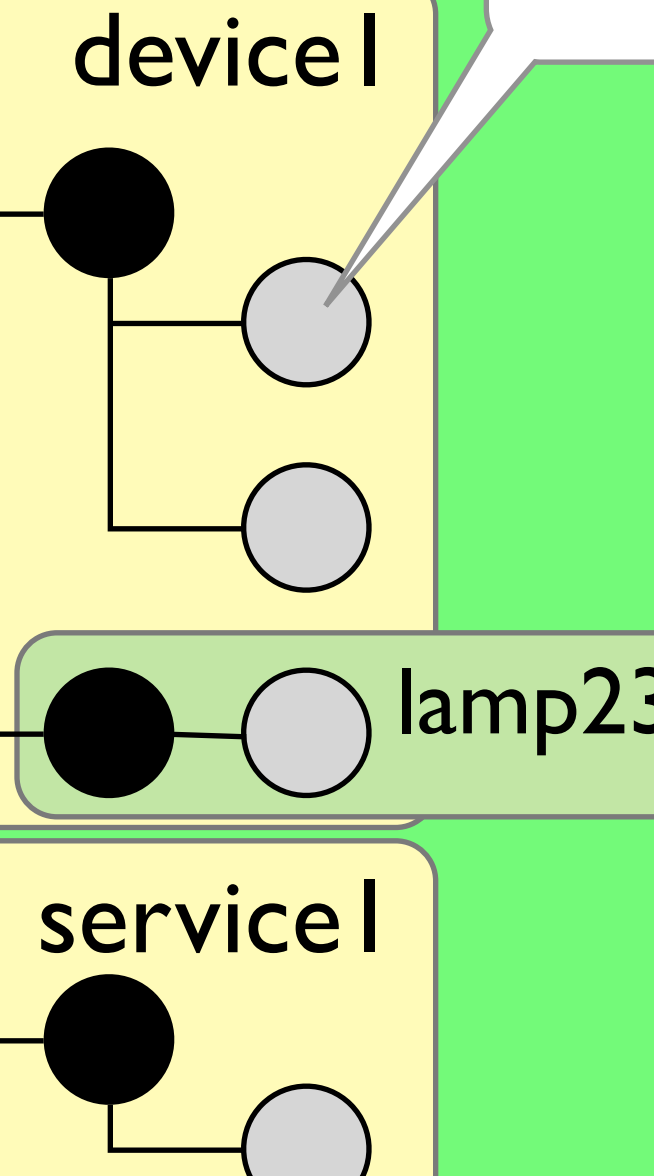
- register
- unregister
- get
- set
- subscribe
- unsubscribe
- registerVirtual
- unregisterVirt.
- search

A Knowledge Agent runs on machines that are powerful enough to host services. It communicates over XML-RPC. Native connectors can easily be implemented.

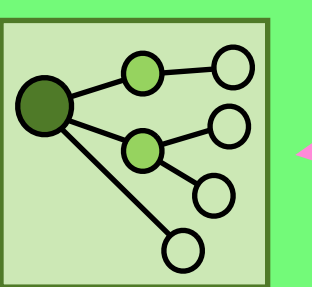
It provides basic functionality for services:

- persistence
- transparent inter-service and inter-machine communication
- access control

Node Attributes:  
Type  
VersionInfo  
AccessRights  
Value



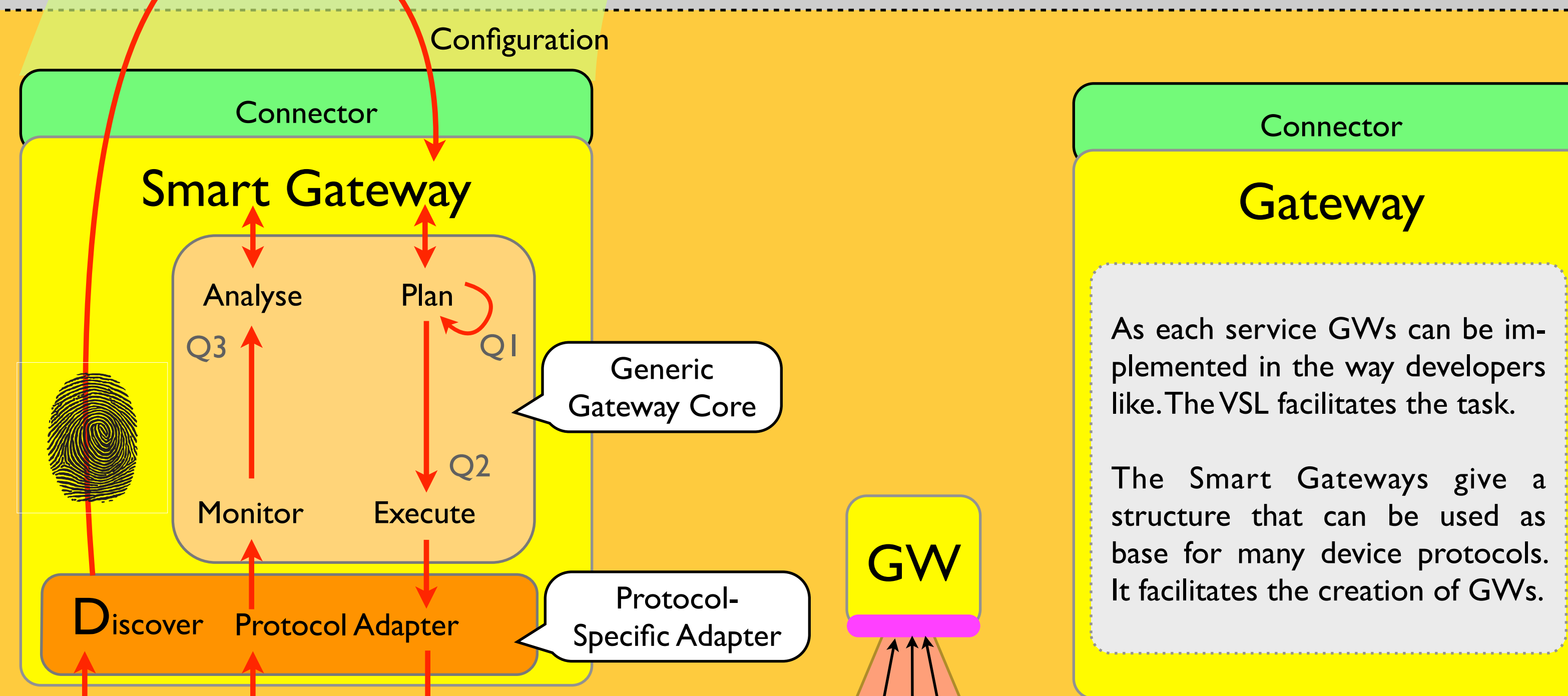
### Model Repository



The Model Repository is the *shared ontology* of the VSL. It stores all service (and device) models as *data types*. A model is the interface description of a service in DS2OS. The abstraction is needed to create *portable services*. The Model Repository is an *open ontology*. Each user can participate uploading new models to the repository. Models can *inherit* from each other, enabling reuse and compatibility. The repository preserves its integrity. *Convergence* of models is essential to. Convergence is realised by tracking the use of models and providing the statistics. This is expected to foster the convergence to certain models for implementing interface compatibility.

### Gateway Services

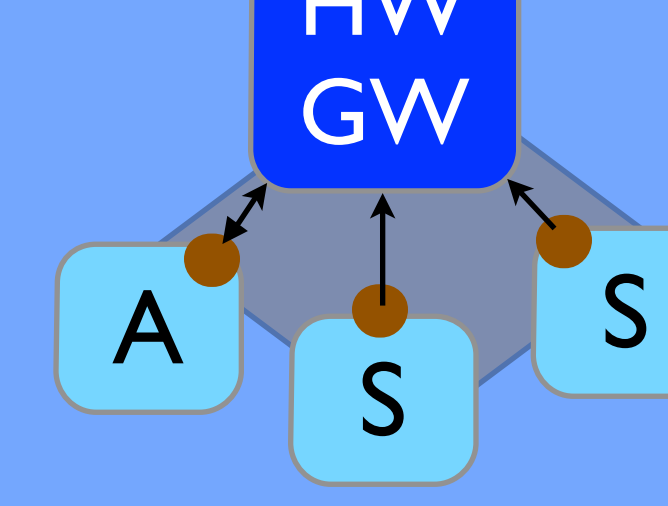
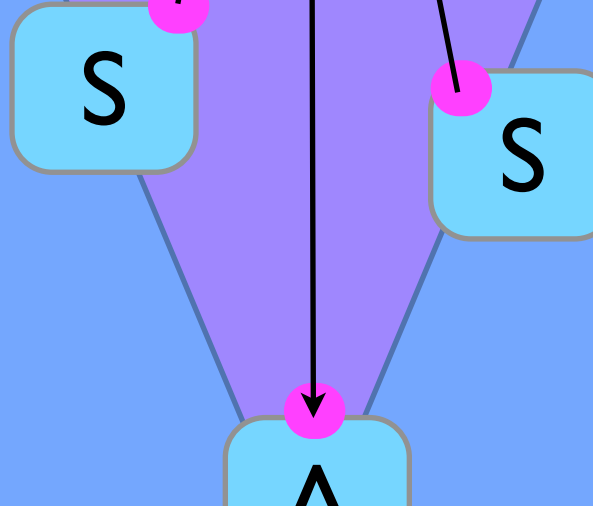
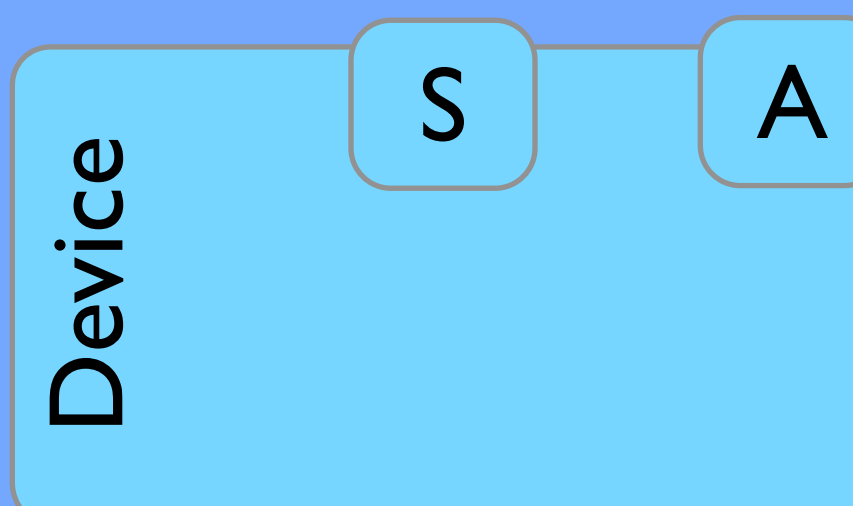
Gateway Services (GW) connect devices with the VSL. They reflect real world changes to VSL device model instances in the and vice-versa. The connection of a GW with a device can happen via special physical interfaces of a machine. The connection of the GW with other devices happens over the VSL. This results in a full logical decoupling of hardware and software making autonomous behavior of gateways necessary.



Like the Model Repository the App Store is open for user created software. For quality control different metrics are implemented. Error reports from smart space instances are collected and evaluated. The number of installed and running copies of a service is tracked and evaluated. Users can rate services. All ratings are made public to ease the user choice for services and to motivate developers to create good software.

### Sensor/ Actuator Devices

The sensors and actuators are the connection between the physical world and the DS2OS system for software-based orchestration. Networked devices for many purposes are available off-the shelf at affordable prices today. The heterogeneity of the communication protocols and abstractions provided by the device vendors makes it necessary to introduce an abstraction layer (VSL) to allow the creation of portable services.



Physical World



<http://www.ds2os.org/>