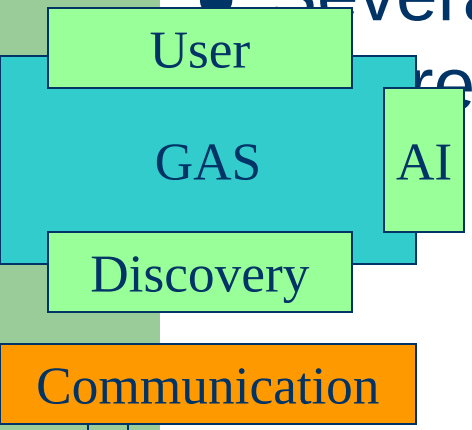
UbiComp Lecture 5

eGadgets – Ontologies

GW Examples

Issues to be addressed

- Several issues are being addressed at two different levels:



eGadget level

Gadgetworld level

… and regarding different aspects:

People usage of eGadgets and Gadgetworlds

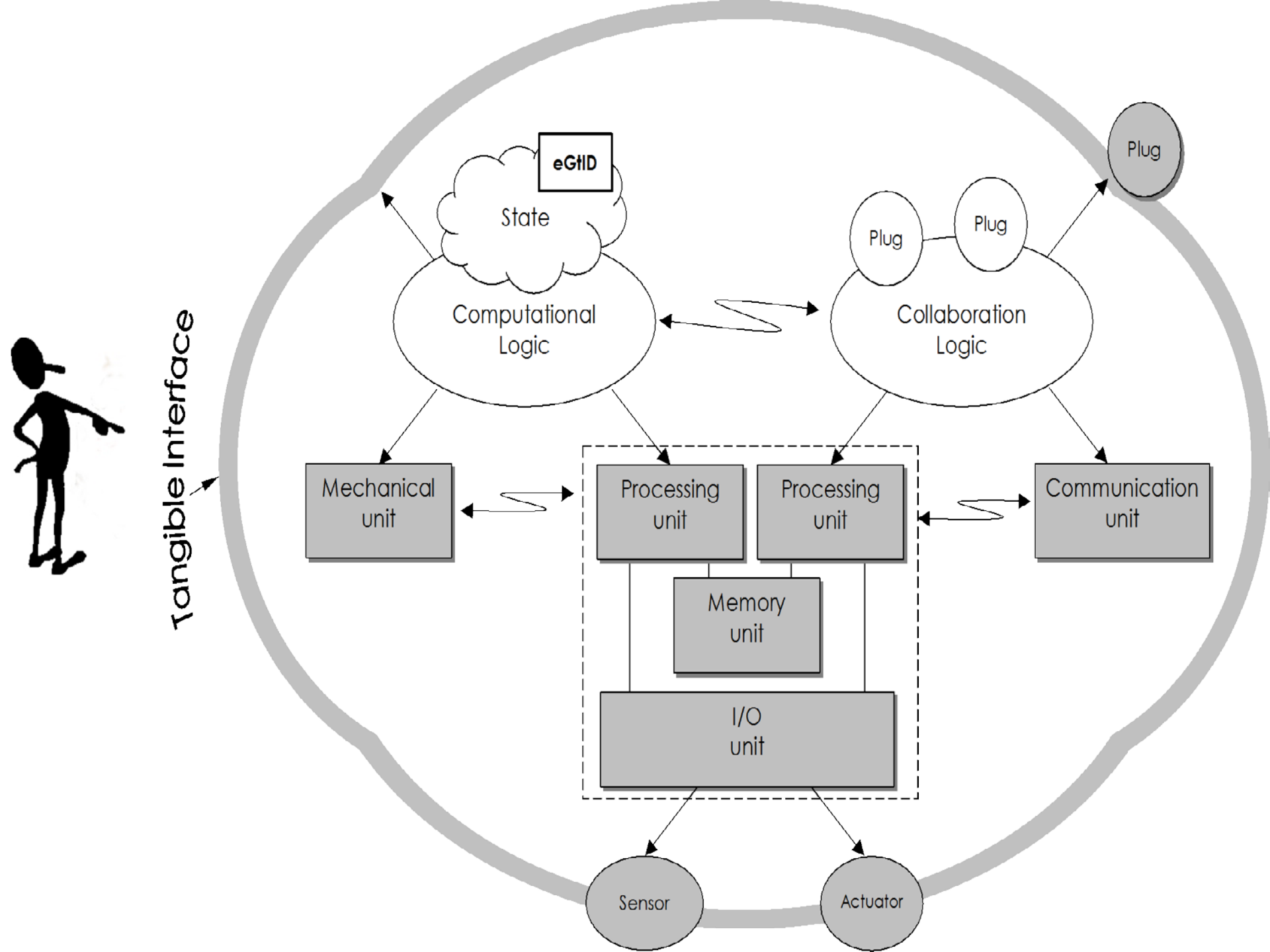
eGadget design

Gadgetworld realization

Role of intelligence

Integration, packaging and miniaturization

Dependence on available and projected technologies


Tangible Interface
eGtID
State
Computational Logic
Collaboration Logic
Plug
Plug
Plug
Mechanical unit
Processing unit
Processing unit
Communication unit
Memory unit
I/O unit
Sensor
Actuator

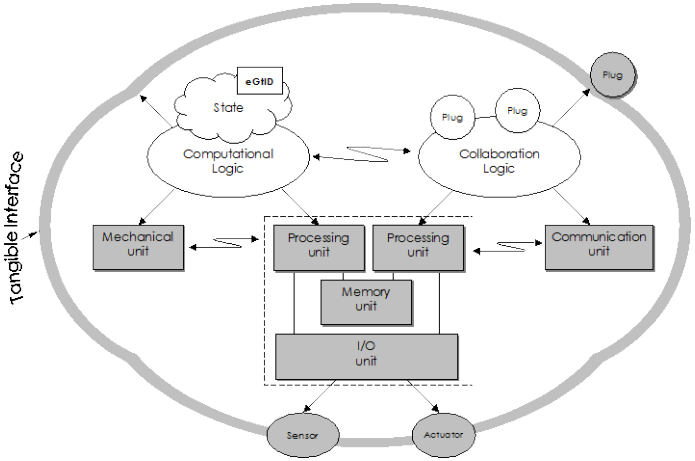
Rule Example (Computational Logic)

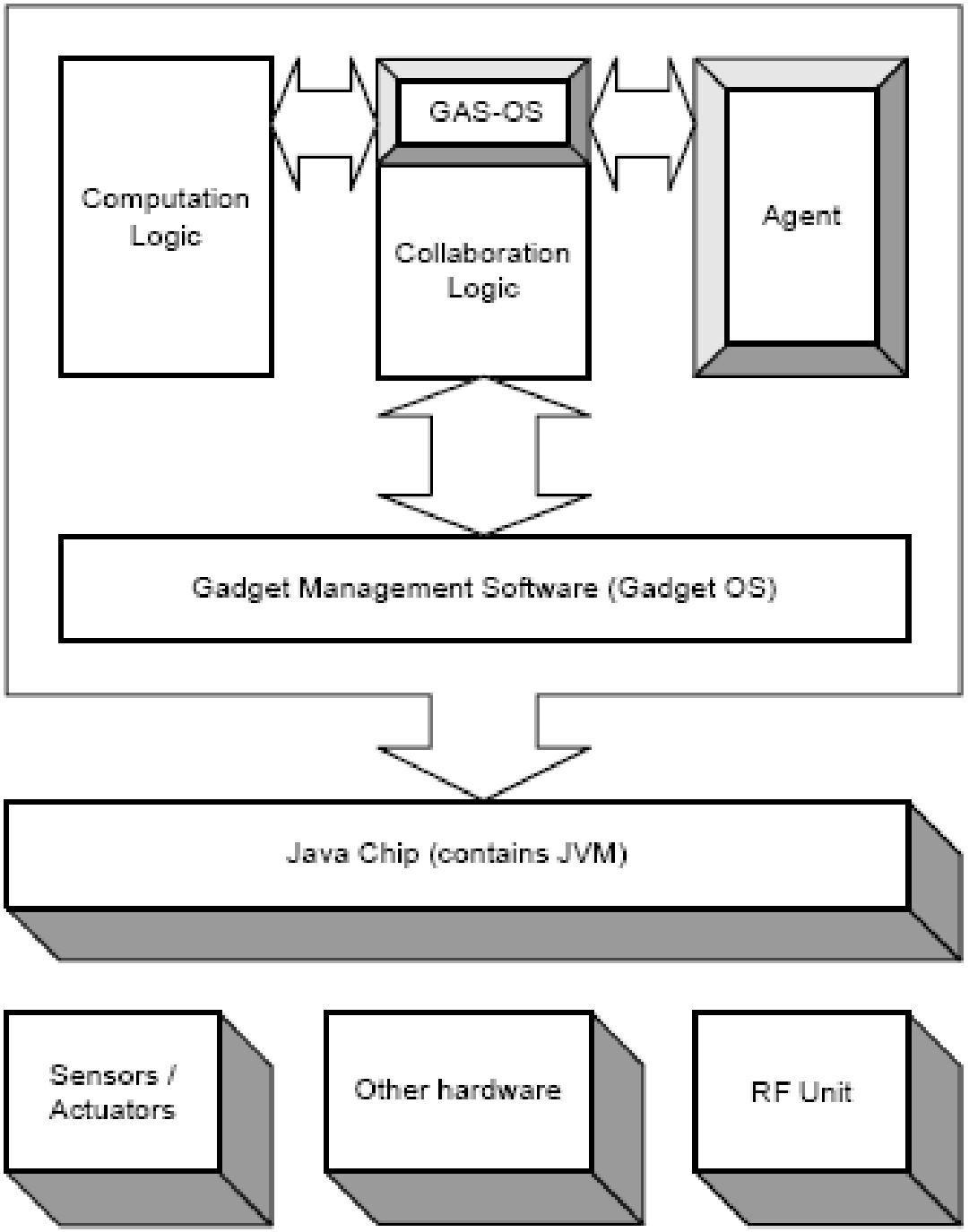
SENSING

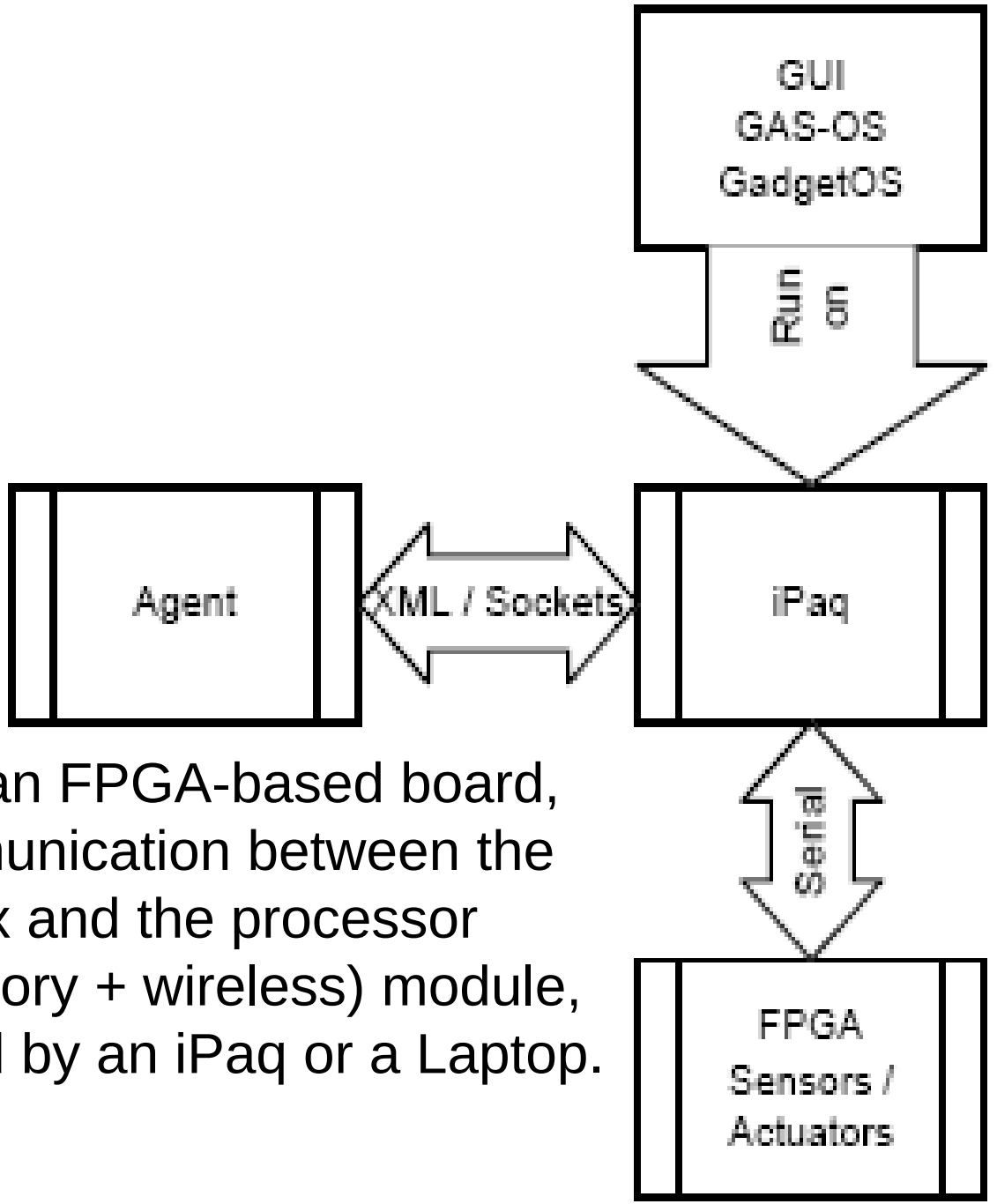
- **When** <the particular CHAIR is NEAR the DESK> **And** <ANY BOOK is ON the DESK> **And** <SOMEONE is sitting on the CHAIR> **And** <The BOOK is OPEN>

ACTUATING

- **Then** <ADJUST the LAMP INTENSITY according to the book LUMINOSITY>.


eGHD
State
Computational Logic
Collaboration Logic
Plug
Plug
Plug
Mechanical unit
Processing unit
Processing unit
Communication unit
Memory unit
I/O unit
Sensor
Actuator
tangible interface


GAS-OS
Computation Logic
Collaboration Logic
Agent
Gadget Management Software (Gadget OS)
Java Chip (contains JVM)
Sensors / Actuators
Other hardware
RF Unit



An eGadget is made of an FPGA-based board, which implements communication between the sensor or actuator matrix and the processor and a (processor + memory + wireless) module, which is currently served by an iPaq or a Laptop.

eGadgets Classes

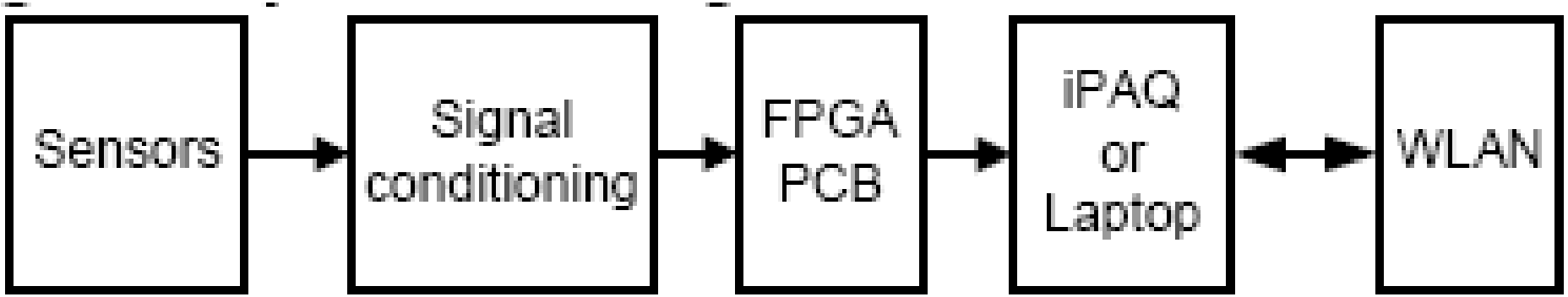
- Sensing eGts
- Actuating eGts.
- Examples of artifacts already implemented include eChair, eDesk, eBook, eLamp etc.

Study eGW

About 10 different objects are now converted to extrovert Gadgets.
A first prototype software tool, the Editor, helps to manipulate e-Gadgets and make synapse links between them.

Sensing eGt Architecture (eBook, eDesk, eChair)

Signal conditioning logic is used to interface the sensors with a Field Programmable Gate Array (FPGA) based circuit, which does some initial preprocessing on the sensor data before sending it serially to the iPAQ or Laptop. The iPAQ/laptop is used to host the GAS-OS, which interprets the serial sensor data. Wireless local area network PCMCIA cards are then used to interconnect the eGts.

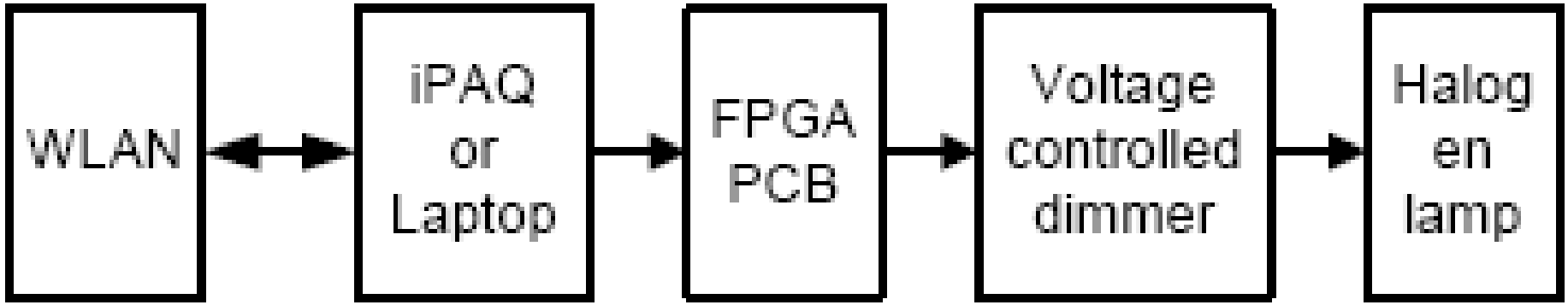


Sensors

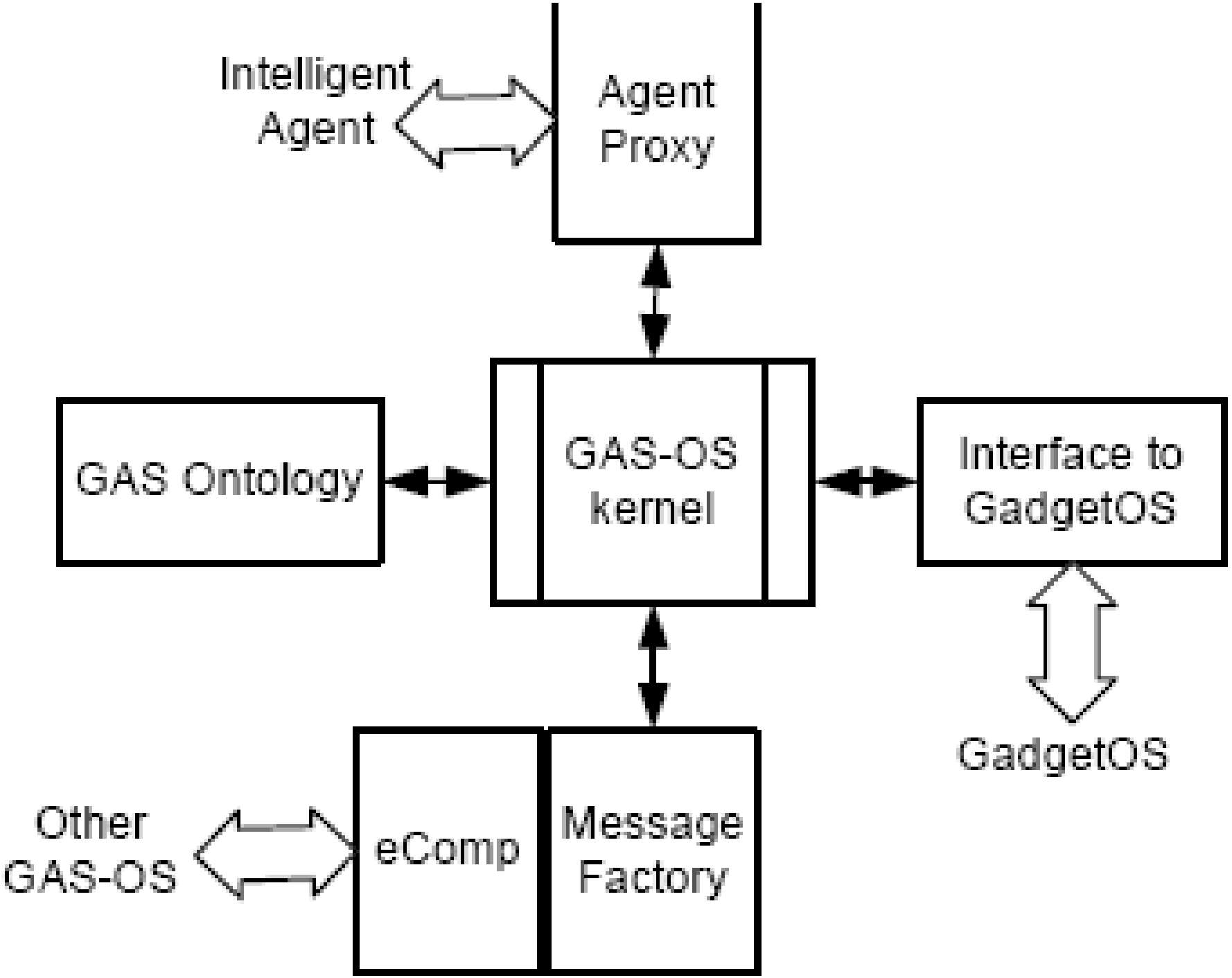
- Pressure pads, **light dependent resistors** (LDRs), bend sensors, ultrasonic transducers and tilt switches.
- The pressure pads are used to detect the presence of a substantial weight such as a person sitting on the eChair
- The LDRs have a dual purpose; firstly for detecting a bigger range of weights such as an object on the eDesk and secondly to detect range of light levels, **the luminosity of the eBook** for example.
- **The eBook has ‘bend sensors’ embedded into the spine to detect whether it is open or closed.**
- An ultrasonic transmitter and receiver pair is used to detect the **proximity of two objects from each other**; the eTable and eChair use this.

eLamp (actuating eGt) Architecture

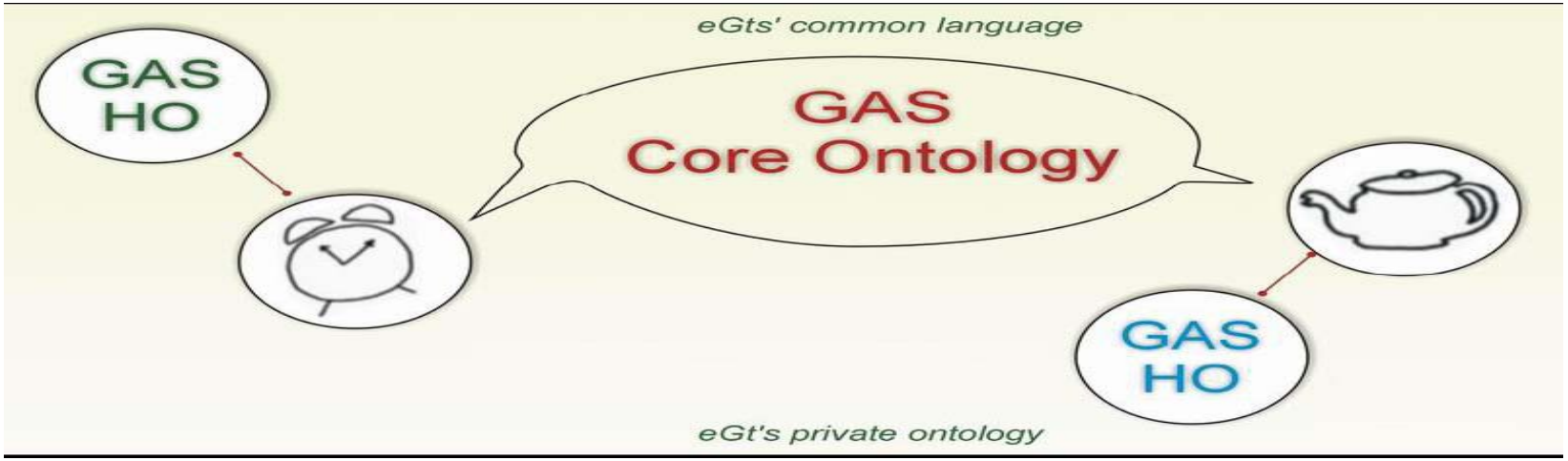
The eLamp receives status information from synapsed eGts over the WLAN and this data is processed by the GAS-OS residing in the iPAQ or the laptop.
Information on the required dimmer level is sent serially to the FPGA PCB which implements pulse width modulation (PWM) to produce an analogue signal whose voltage level is proportional to the desired light level. The analogue signal is the input to a voltage-controlled dimmer, which is required to provide the higher voltages necessary to dim the eLamp.



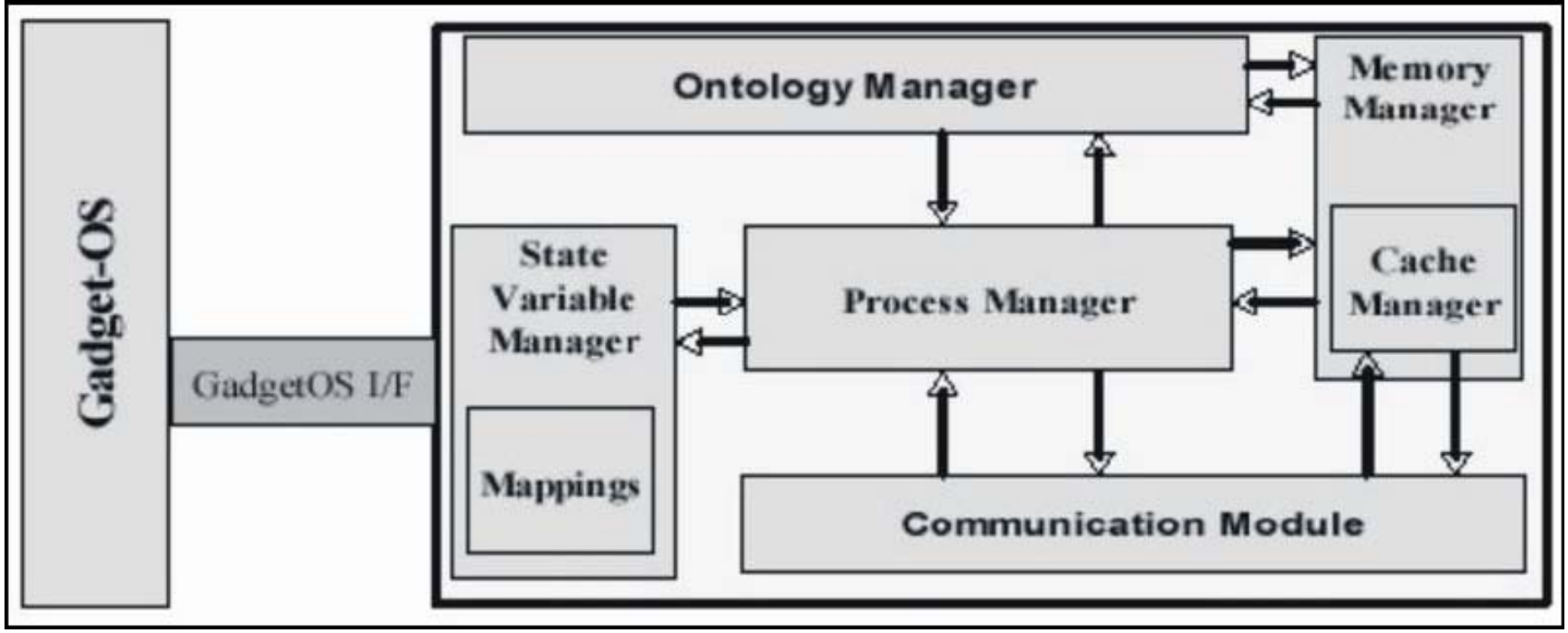
GAS-OS ARCHITECTURE



- ***GAS Ontology*** provides the **common language for the communication and collaboration among eGts**
- Describes the semantics of the basic terms (eGt, Plug, Synapse, eGW) and Defines the relations among them
- GAS Ontology ensures eGts replacement feasibility, Plugs compatibility, Services discovery



- GAS ontology consists of two layers:
- ***GAS Core ontology*** (GAS-CO)
- • describes the semantics of the basic terms
- • defines their relations and their roles
- • provides the **common language among eGts**
- ***GAS Higher ontology*** (GAS-HO)
- • contains eGt acquired knowledge



GAS - operating system manages the resources of eGts and enables participation in eGWs through plug and synapse management services

Ontology Manager is responsible for the interaction of the eGt with its stored ontology and the management of this ontology

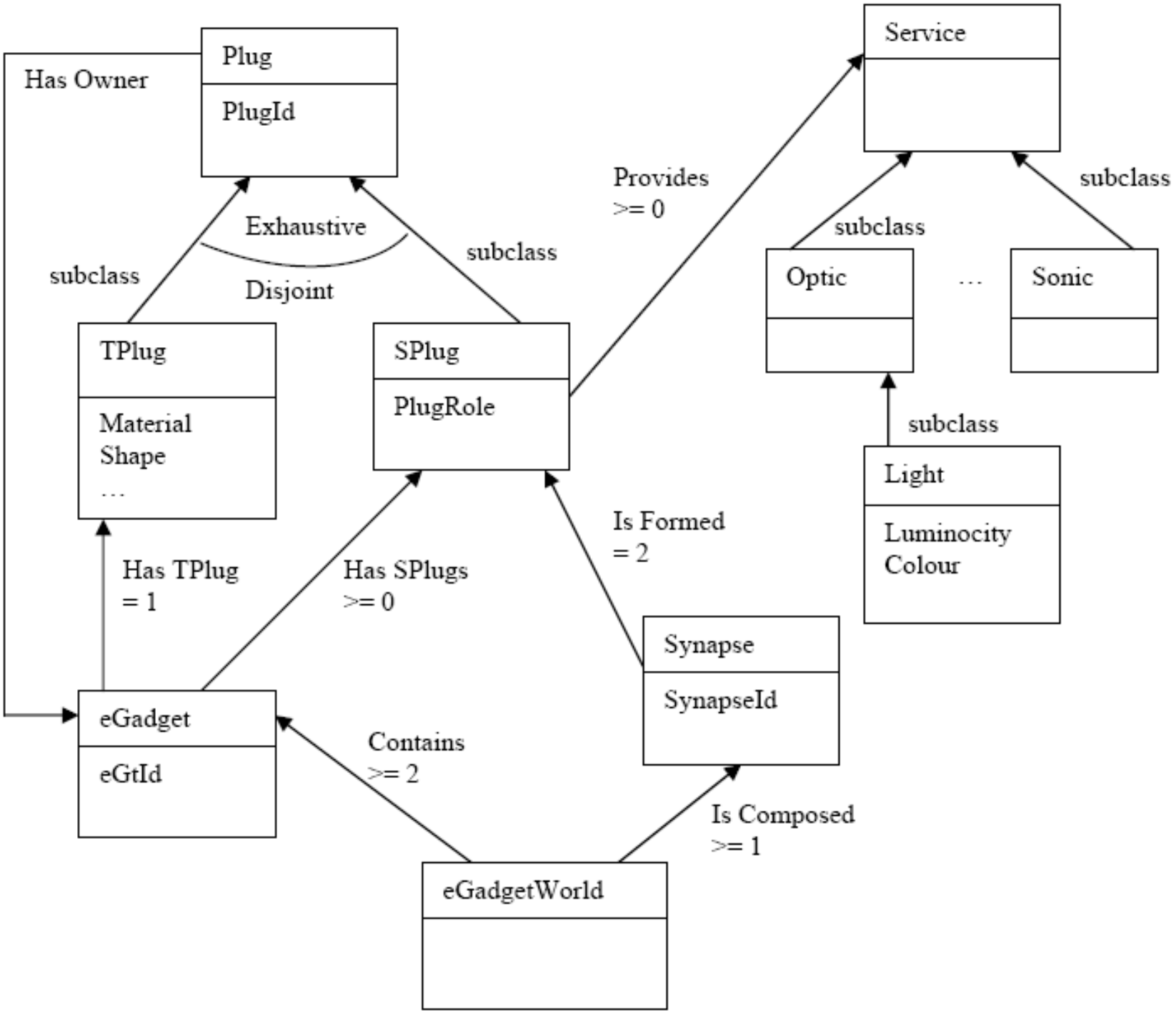
- Edits GAS-HO, making feasible the “storage” and “abstraction” of knowledge
- Enables the exchange of knowledge from eGts’ ontologies
- Composes eGts’ queries

GAS Ontology

- Describes the hierarchy of basic concepts and the local eGt capabilities and experience.
- It is encoded in XML; it contains a set of basic terms that are understandable by all eGts and a mechanism to translate local definitions using the basic terms.
- It is used to ensure semantic compatibility between different eGts.

GAS-CO

- Is the common language of eGts, so it must describe the semantics of the basic terms of eGWs and define their inter-relations.
- Contains the service classification to support the service discovery mechanism.
- **Defines the rules for plugs compatibility and eGts replaceability.**
- Describes the language that eGts use to communicate and/or collaborate,
- **Major goal of GAS-CO is to contain only the necessary information in order to be very small. Thus even eGts with limited memory capacity can store it.**
- Key Issue: **GAS-CO cannot be changed either from the manufacturer of an eGt or from an eGadget user, so it is static.**


Plug
PlugId
Has Owner
Service
Provides
>= 0
subclass
Exhaustive
Disjoint
subclass
subclass
subclass
Optic
…
Sonic
TPlug
Material
Shape
…
SPlug
PlugRole
subclass
Light
Luminocity
Colour
Is Formed
= 2
Has TPlug
= 1
Has SPlugs
>= 0
Synapse
SynapseId
eGadget
eGtId
Contains
>= 2
Is Composed
>= 1
eGadgetWorld

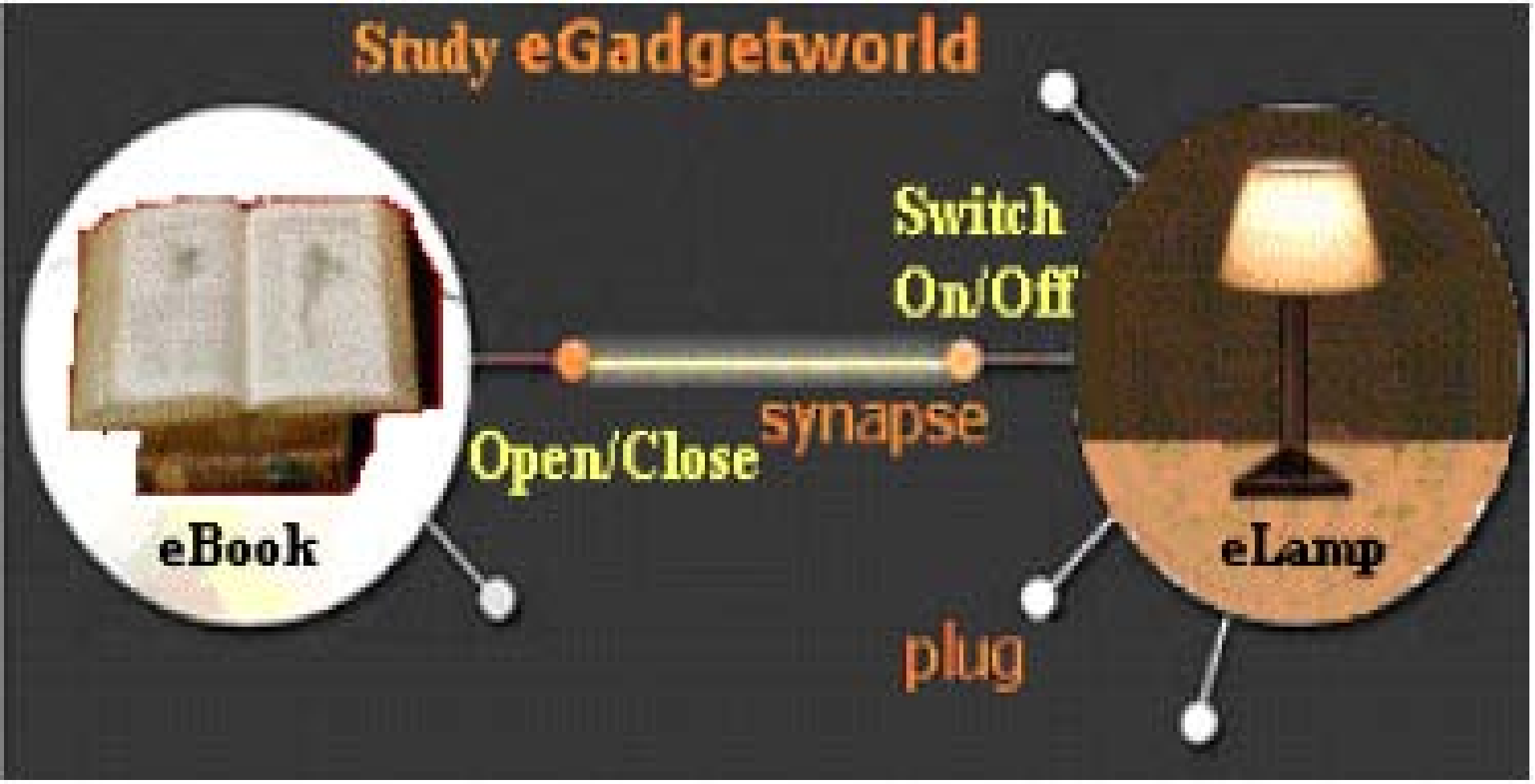
GAS-HO

- eGt’ s private ontology
- Can be “unlimited” and depending on eGt’s memory capacity.
- Can be changed over time without causing problems to eGts communication.
- Represents both the description of an eGt and its acquired knowledge.
- These descriptions follow the definitions contained in the GAS-CO. The knowledge represented by GASHO is described as instances of the classes defined into the GAS-CO.
- Note that the GAS-HO is not a stand-alone ontology, as it does not contain the definition of its concepts and their relations.

*GAS-HO-**static***

- Represents the description of **a particular eGt** containing information about eGt's plugs, the services that are provided through these plugs, its sensors and actuators, as well as its physical characteristics.
- Description of an eGt is presented as a number of **instances of the concepts that are defined in GAS-CO**.

Example


Study eGadgetworld
Switch
On/Off
synapse
Open/Close
eBook
eLamp
plug

Plugs of Study GadgetWorld

Desk	Lamp	Book	Chair
T-plug O	T-plug O	T-plug O	T-plug O
Weight O	On/off I	Open / closed O	Occupancy O
Proximity O	Intensity I	Luminosity O	

Key: O=output plug,
I=input plug,
H=Higher level
(composite) plug).

eLamp GAS-HO static

- Contains the **knowledge about the physical properties** of “eLamp”, such as its **luminosity**, the **description** of its **SPlug “switch on/off”** based on the definition provided by **GAS-CO**, as well as the **declaration** that the SPlug “switch on/off” provides the service “light”

eBook’s GAS-HO-static

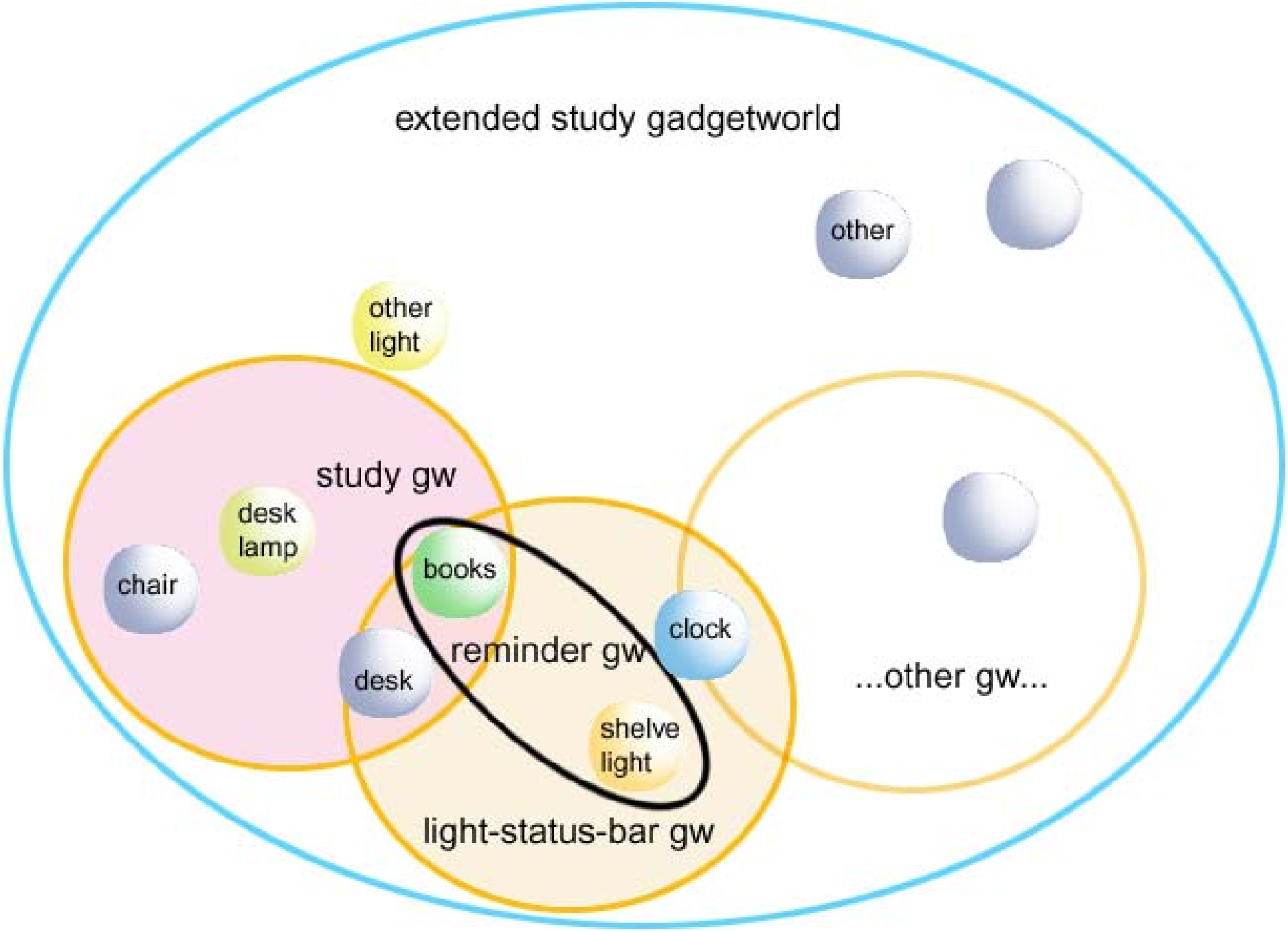
- ontains the description of SPlug “open/close”.

*GAS-HO-**volatile***

- Contains information derived from the eGt's **acquired knowledge** and its use.
- Describes the **synapses** which the eGt's plugs are connected to, the **eGWs** which it takes part to, as well as information about the **capabilities of other artifacts that** has acquainted through communication.
- An eGt's GAS-HO-volatile is updated during the eGt's various activities.

Synapses

- Desk (weight=1) -> Lamp (on/off=on)
- Chair (occupancy=1) -> Lamp (on/off=on)
- Book (open/closed=open) -> Lamp (on/off=on)
- Book (luminosity) -> Lamp (intensity)


extended study gadgetworld
other
other light
study gw
desk lamp
chair
books
reminder gw
clock
desk
shelve light
light-status-bar gw
...other gw...

GAS-HO Volatile

- For example, when the SPlug “open/close” of the “eBook” establishes a Synapse with the SPlug “switch on/off” of the “eLamp”, both these eGts store the **description of this Synapse** into their GAS-HO-volatile.
- Knowledge emerged from this synapse is stored in the GAS-HO-volatile **of both the eGts that participate in it**.

Example

- So the eBook “knows” that it’s SPlug “open/close” participates to a synapse with an SPlug that provides the service “light” with a specific luminosity.
- If this **synapse is broken**, for example because of a failure at the eLamp, a new eGt having an SPlug that provides the service “light” must be found.

Example (cont.)

- The eBook's GAS-OS needs to find another eGt with an SPlug that provides the service "light".
- The eBook's GAS-OS is responsible to send a message for ***service discovery*** to the other eGts' GAS-OS that participate in the same eGW.
- This type of message is **predefined** and contains the **type of the requested service** and the **service's attributes**.

Service Discovery

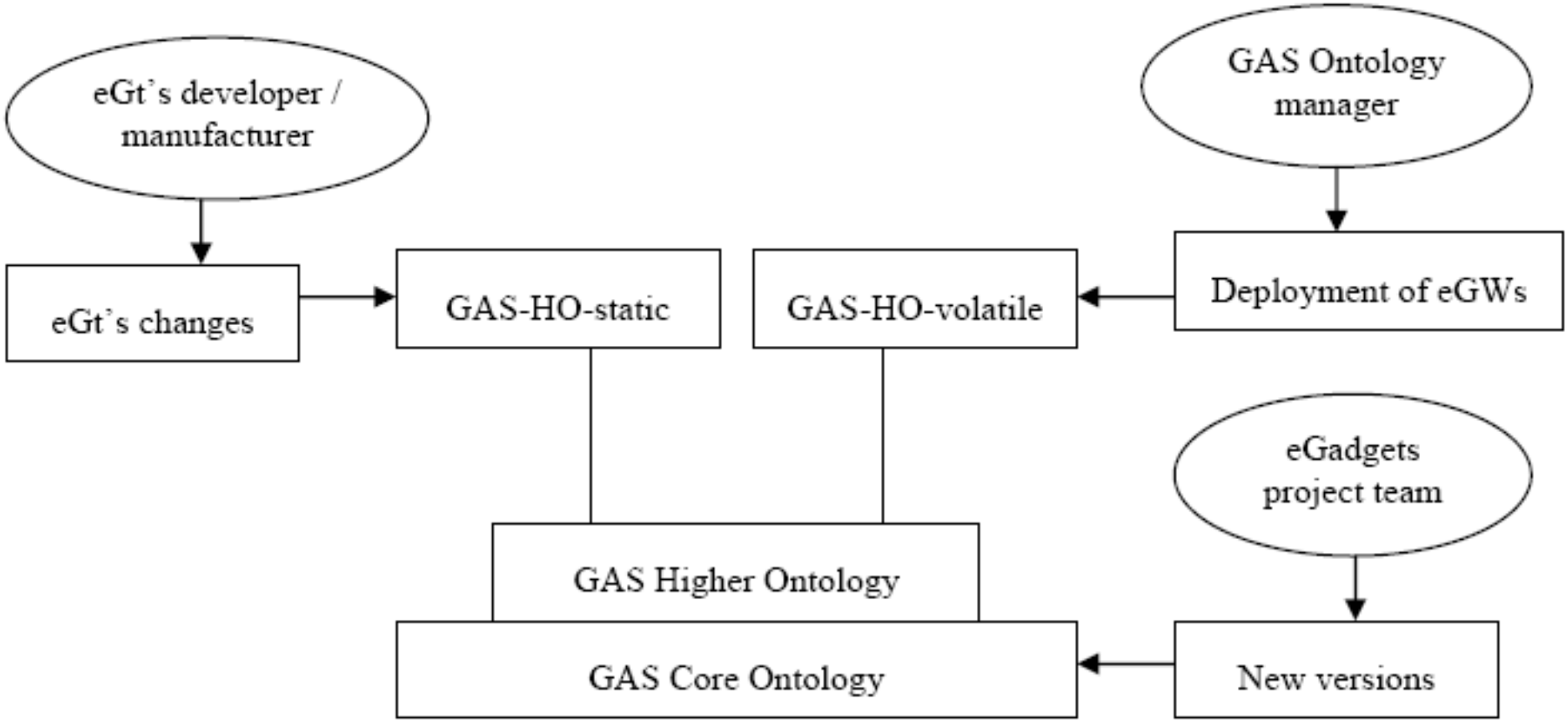
- When the GAS-OS of an eGt receives a service discovery message, forwards it to the eGt's GAS Ontology Manager.
- Assume that the eGt "eDeskLamp" participates in the study eGW and that this is the first eGt that gets the message for service discovery.
- The eDeskLamp's GAS Ontology Manager first queries ***GAS-HO-static*** of eDeskLamp in order to find if this eGt has an SPlug that provides the service "light".

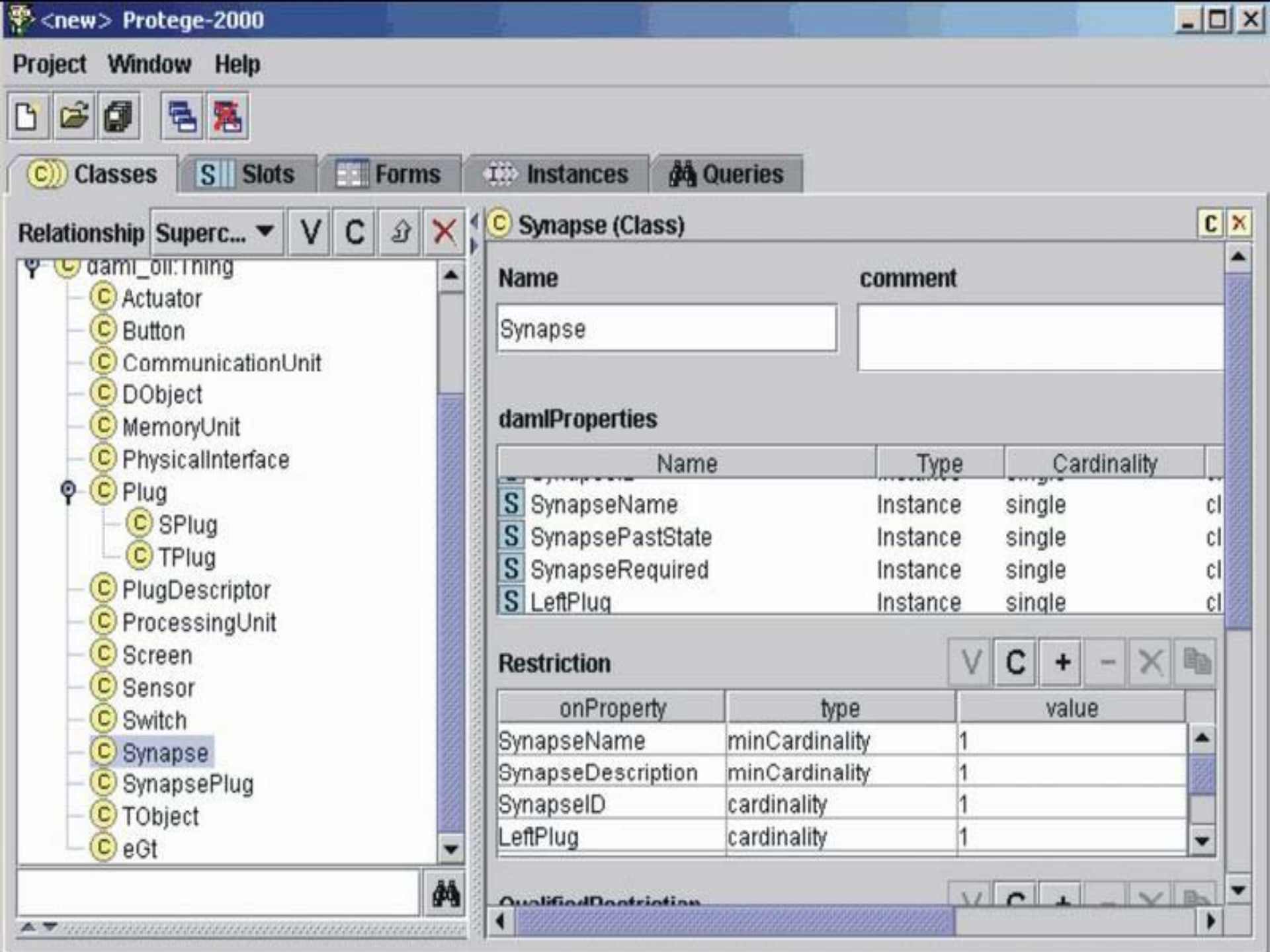
Service Discovery (cont)

- If we assume that the eDeskLamp has the SPlug “LampDimmer” that provides the service light, the GAS Ontology manager will send to the eDeskLamp’s GAS-OS a message with the description of this SPlug.
- If such an SPlug is not provided by the eDeskLamp, the GAS Ontology Manager queries the eDeskLamp’s ***GAS-HO-volatile*** in order to find if another eGt, with which the eDeskLamp has previously collaborated, provides such an SPlug.
- If the GAS Ontology Manager finds into

- If the queried eGt, in our example the eDeskLamp, has no information about an SPlug that provides the requested service, the control is sent back to GAS-OS, which is responsible to send the query message for the service discovery to another eGadget.
- Note that all the eGadgets have the same service classification, which is stored into the GAS-CO; thus the messages for

Lifecycle of GAS Ontology




<new> Protege-2000
Project Window Help
Classes Slots Forms Instances Queries
Relationship Superc...
daml_oil:Thing
Actuator
Button
CommunicationUnit
DObject
MemoryUnit
PhysicalInterface
Plug
SPlug
TPlug
PlugDescriptor
ProcessingUnit
Screen
Sensor
Switch
Synapse
SynapsePlug
TObject
eGt
Synapse (Class)
Name
comment
Synapse
damlProperties
Name
Type
Cardinality
SynapseName Instance single
SynapsePastState Instance single
SynapseRequired Instance single
LeftPlug Instance single
Restriction
onProperty
type
value
SynapseName minCardinality 1
SynapseDescription minCardinality 1
SynapseID cardinality 1
LeftPlug cardinality 1

Advantages of this approach

- Can scale both “upwards” (towards the assembly of more complex objects, i.e. from objects to rooms, up to buildings, cities and so on) and “downwards” (towards the decomposition of eGts into smaller parts, i.e. towards the concept of ‘smart dust’).
- Moreover it treats evenly different objects, that may range from powerful ones (having their own processing and communication), to very minimal ones (that can be considered as tagged artifacts, which borrow processing and storage capabilities from servers for example).
- In this sense it is quite a ‘democratic’ system for extrovert gadgets.