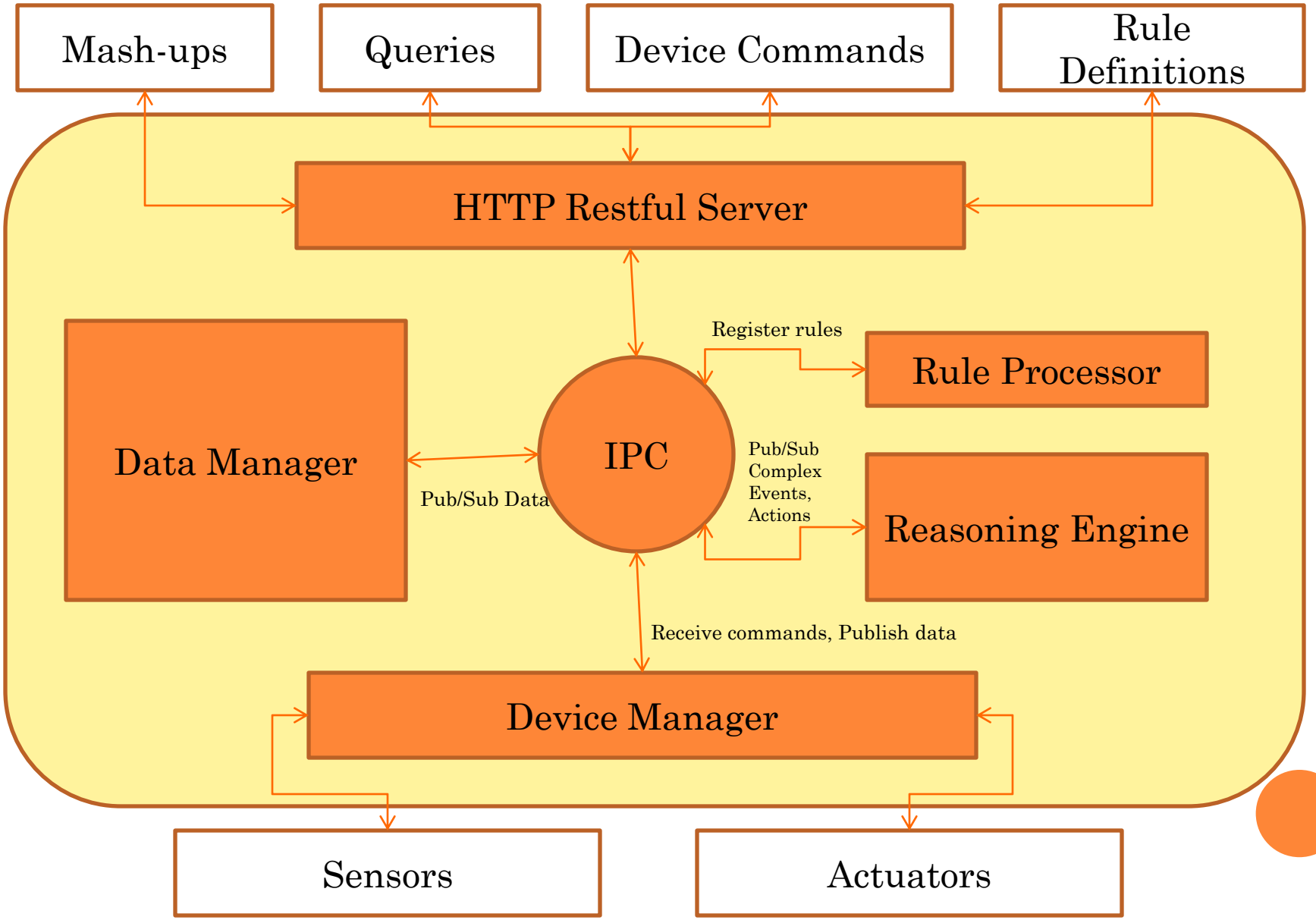
SMARTGATEWAY

- Dr. Yann-Hang Lee
- Shankar Nair

4 Basic Components

- Device Manager
  - Low level backend
- Http Restful Server
  - High level frontend
  - User explicit control
- Data Manager
  - Streaming data storage
  - Allows subscribing/publishing data
- Reasoning Engine
  - Detect temporal complex events
  - Execute actions based on defined rules

Smartgateway Architecture



Device Manager

- Main Purpose
  - Handle communication with physical devices.
  - Publish device-status changes to higher layers.
  - Receive commands that must be run on the physical device. These commands can come from:
    - HTTP Rest Server
    - Reasoning Engine
  - Exposes a device-instantiation server to handle dynamic device creations

Device Manager - Architecture

- DeviceBase
  - This is an abstract class that models the communication with a physical device such as its read/write frequency, and lays out a framework so that upper layers need only implement device specific functionality and do not care about the actual communication with the device.
  - It also takes care of multithreading and error handling.

Device Manager - Architecture

- Device
  - The template class inherits from the DeviceBase and adds the capability of adding a custom handler.
  - E.g. int for FDs or CvCapture for OpenCV programs
- DeviceDbus
  - Every physical device is represented by a Dbus object.
  - Implements interface as specified in the interface-xml file.

HTTP_REST_SERVER

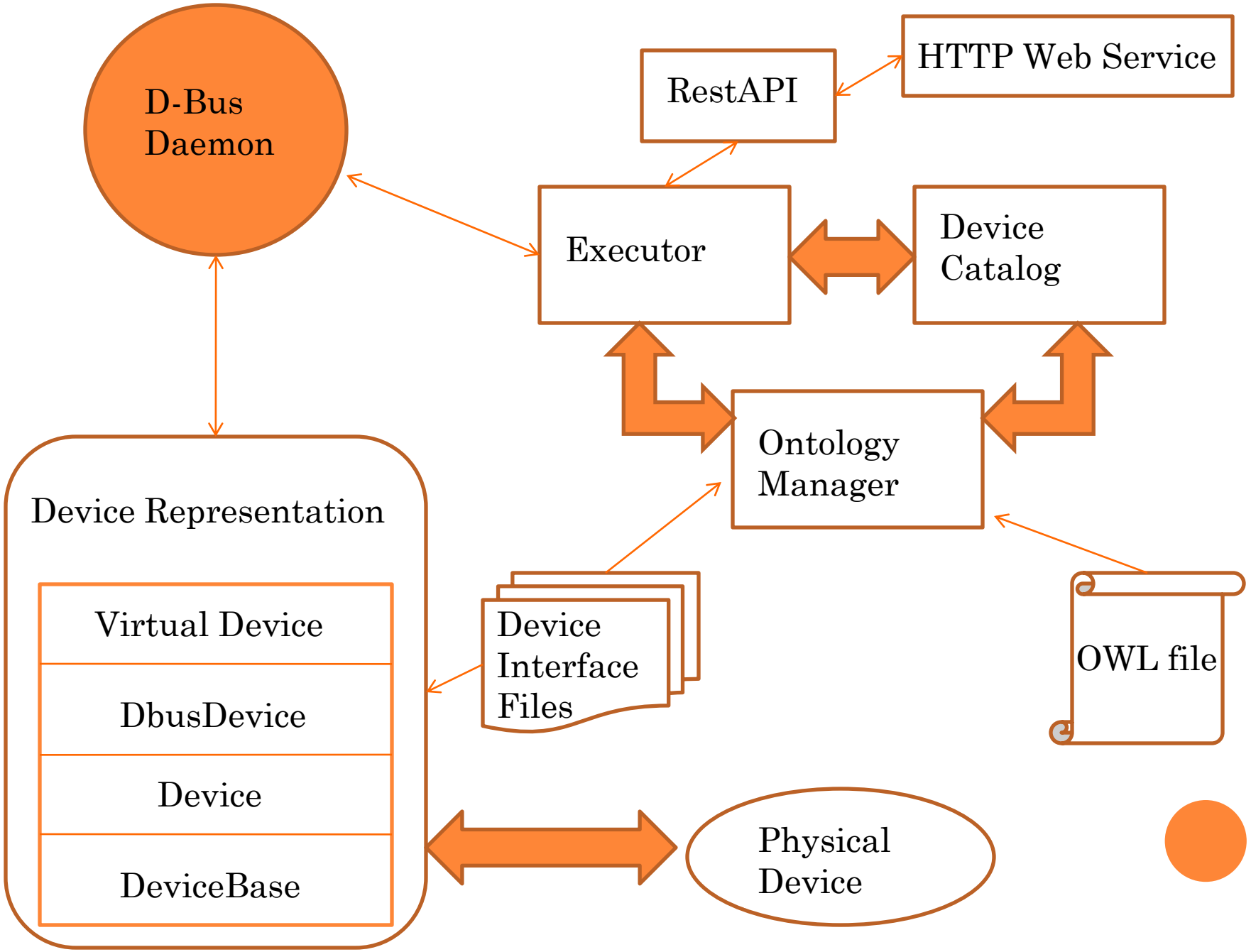
- Main purpose
  - Expose RESTful web-interface.
  - Maintains catalog of devices.
  - Provides framework to access functionality of devices.
  - Provides basic query capabilities to lookup devices.
  - Handles actual instantiation of devices by contacting the DeviceManager.
  - Provides view of data and supports mashups (TBD)

HTTP_REST_SERVER – ARCHITECTURE

- Httphandler
  - Implements HTTP web-server handling.
- RestAPI
  - Provides necessary framework to register Rest APIs.
- Executor
  - Implements RestAPIs.
  - Formats response output.

HTTP_REST_SERVER - ARCHITECTURE

- OntologyManager
  - Provides APIs to support querying of devices.
  - Processes OWL file to initialize the DeviceCatalog
- DeviceCatalog
  - Maintains map of all active devices.
  - Provides APIs for invoking methods on the devices.
  - Processes interface-xml to extract interface information.
  - Associates physical devices to their respective interface based on the Ontology.


D-Bus Daemon
RestAPI
HTTP Web Service
Executor
Device Catalog
Ontology Manager
Device Representation
Virtual Device
DbusDevice
Device
DeviceBase
Device Interface Files
OWL file
Physical Device

Libraries

- Glib-2.0 (-lgobject-2.0 -lglib-2.0)
  - Used for the internal representation of a D-Bus Object using Glib.
- D-bus (-ldbus-glib-1 -ldbus-1)
  - Used for D-bus support.
- Microhttpd (-lmicrohttpd)
  - Used to support the HTTP server implementation.
- OWLCPP (-lowlcpp_io -lowlcpp_logic -lowlcpp_rdf)
  - Used to read, parse and query OWL ontology files within program.
- Raptor (-lraptor)
  - Used by OWLCPP to parse the Ontology file.
- XML (-lxml2)
  - Used by OWLCPP to read the Ontology file.
- FaCT++ (-lfactpp_kernel)
  - Used by OWLCPP to support DL reasoning within the program.
- Boost (-lboost_filesystem -lboost_program_options -lboost_system)
  - Used to ease file-io, string handling and various HTTP response output formating such as JSON, XML etc.

Thank you!