TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Institut für Computertechnik
Institute of Computer Technology

A Multi-Agent-Based Middleware for the Development of Complex Architectures

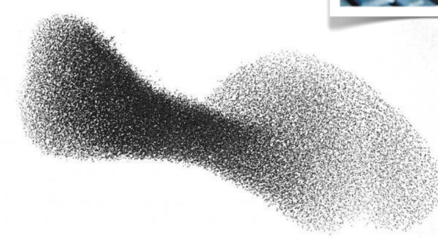
Alexander Wendt, Stefan Wilker, Marcus Meisel and Thilo Sauter

13.06.2018

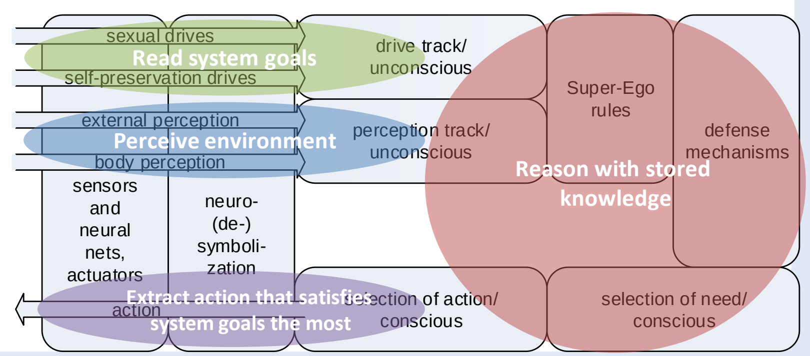
Usage of Complex Systems

ACONA
https://github.com/aconaframework/acona

- Smart Grids
  - Electric grid with additional IT infrastructure
  - Connect heterogenous hardware with algorithms
- Multi-Agent-Systems
  - Each agent – Simple behavior
  - Multi agents – Emergent behavior



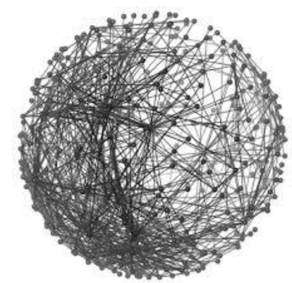
- Cognitive Architectures
  - Mimic decision-making in biology
  - Selection and prioritization tasks
  - Artificial life



ACONA
https://github.com/aconaframework/acona

Design of Complex Systems

- Requirements on complex architectures
  - Handle heterogeneity: E.g. several types of hardware from different producers
  - Handle complexity: Interactions between different components
  - Handle adaptability: Extensions and change requests

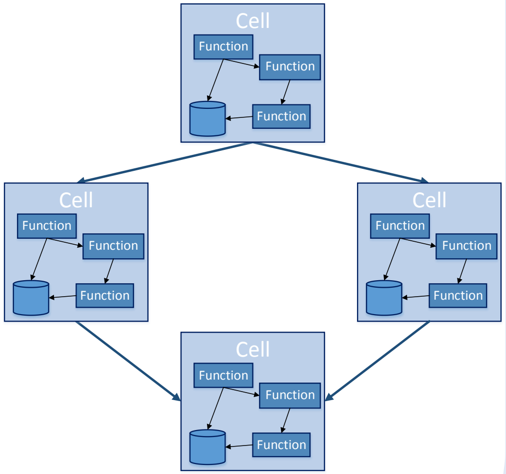


- Solution: Middleware
  - Connect disconnected systems through an application
  - Hide communication infrastructure
  - Use common interface for applications

Problem Statement

- Goal: Implement cognitive systems in industrial domain with one middleware

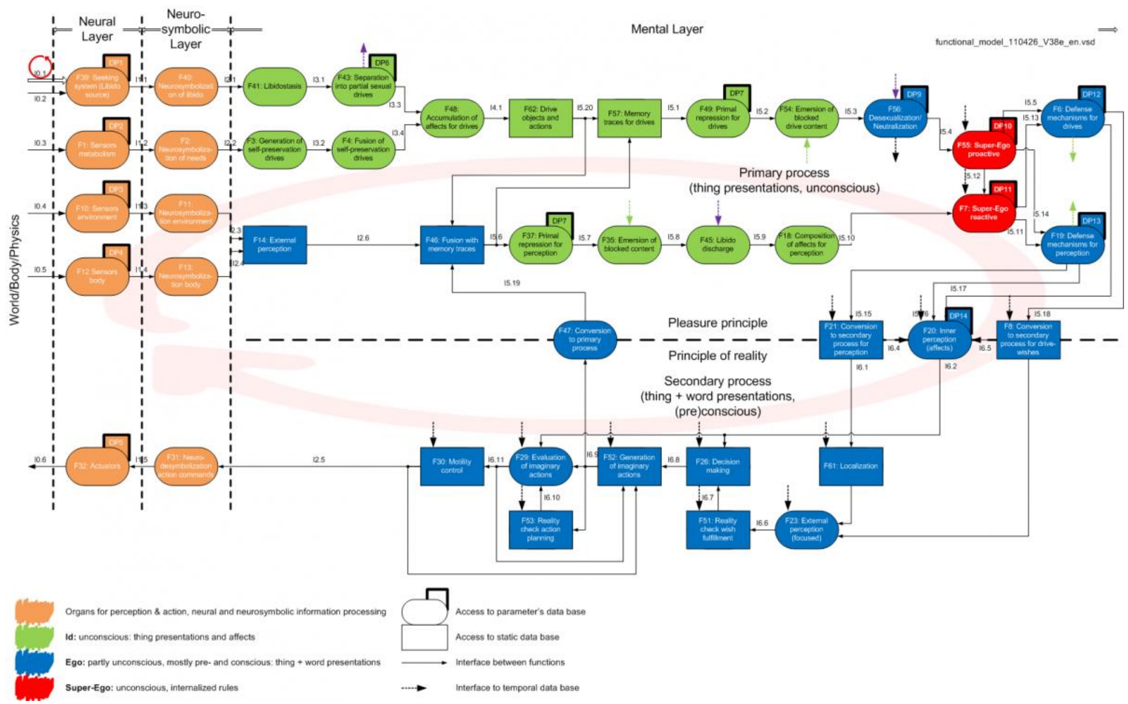
- Requirements
  - Agent system with „body“ and encapsulated internal functions
  - Necessary infrastructure provided by internal functions
  - Flexible enough to support a cognitive architecture



Validation with a Cognitive Architecture

ACONA
https://github.com/aconaframework/acona

- Large collection of interacting modules and memories
- Hard to develop
- Research->often updated



- If it is possible to implement a cognitive architecture, then it is possible to implement it for industrial applications too

Current Solutions

- Smart Grids Middleware
  - OpenMuc: No agent system, limited value range on channels
  - ZeroMQ: Each client needs port -> limited number of agents/functions
  - MQTT, e.g. RabbitMQ: Missing function infrastructure -> e.g. Request/Response-pattern
- Java JADE
  - Multi-agent system with agent and behaviors -> Good
  - Missing function infrastructure -> lots of repeated programming
  - Behavior structure not parallelizable -> no blocking functions

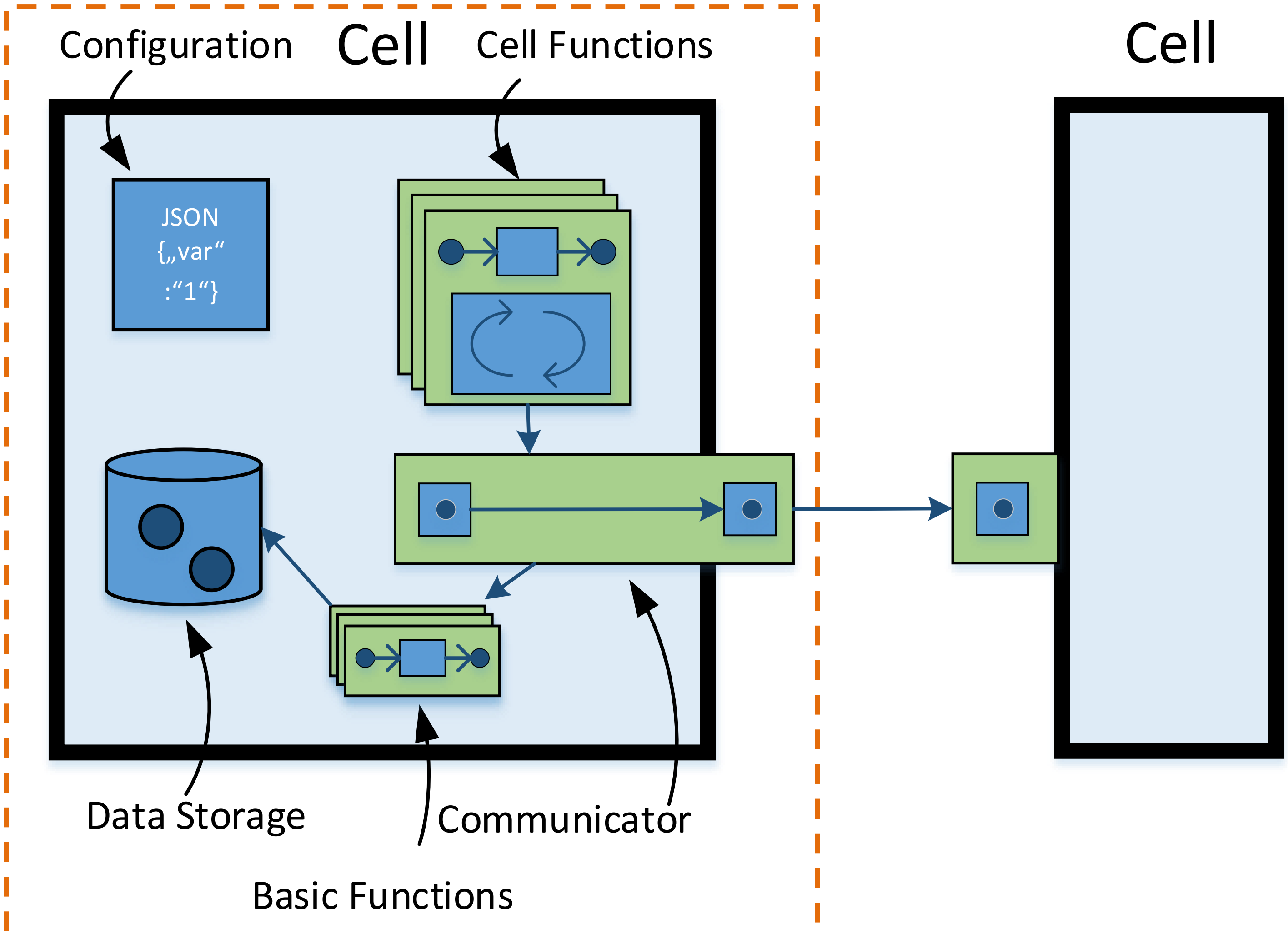
Approach

- Java JADE as base
  - Use existing agent concept
  - Use existing communication methods (FIPA)
- Add extra infrastructure layer to Java JADE
  - Thread-based functions instead of behaviors
  - Function dependencies in data instead of direct references
  - Connection to JUnit to inject data

- Framework A Multi-Agent-Based Middleware for the Development of Complex Architectures (**ACONA**)

Functionality of a Cell

ACONA
https://github.com/aconaframework/acona



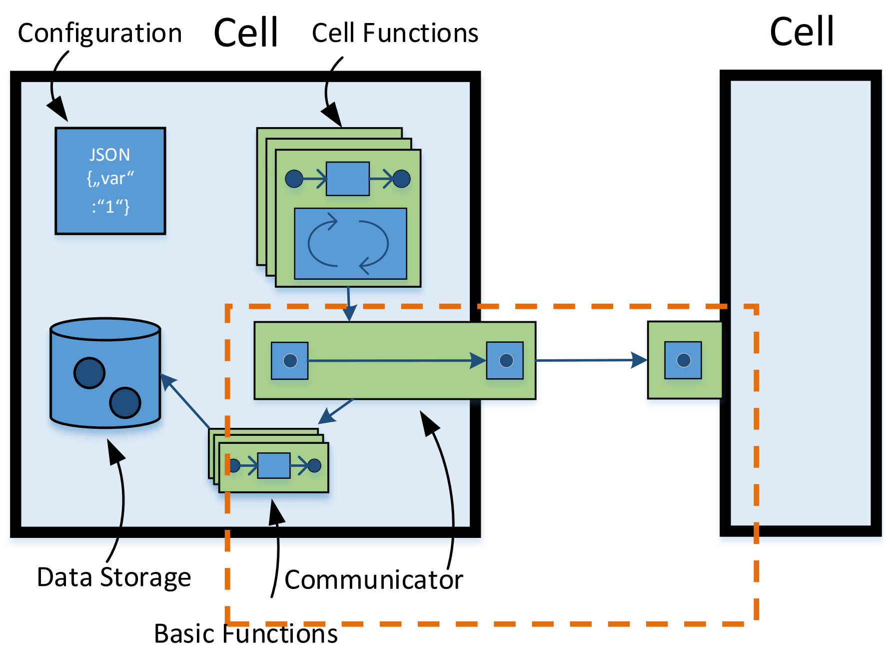
Communicator

ACONA
https://github.com/aconaframework/acona

- Communication between functions internally and between cells

- Message-based

- Remote procedure call in any other function or cell



Data Storage

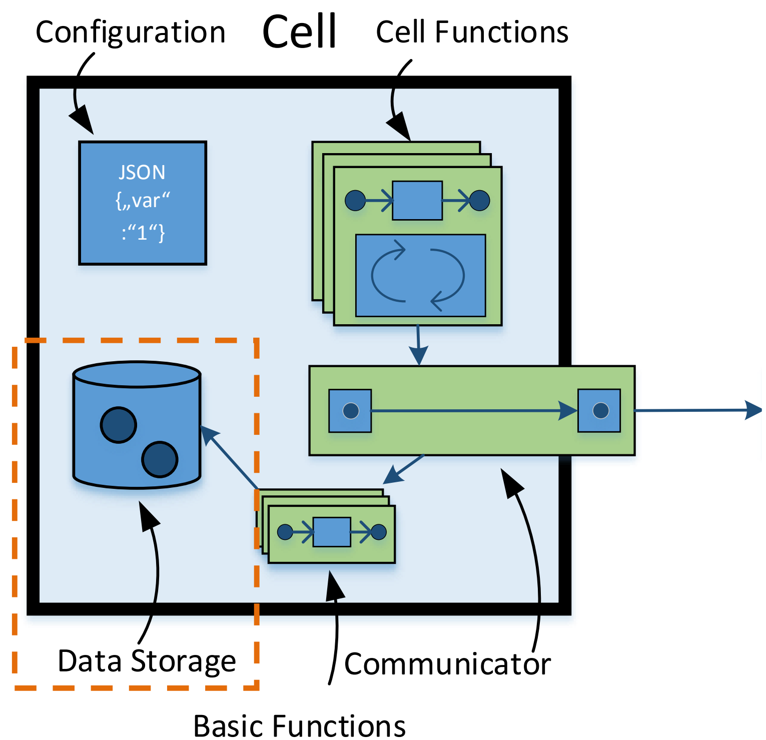
- Key-Value storage

  Datapoint-based

- JSON based data structures

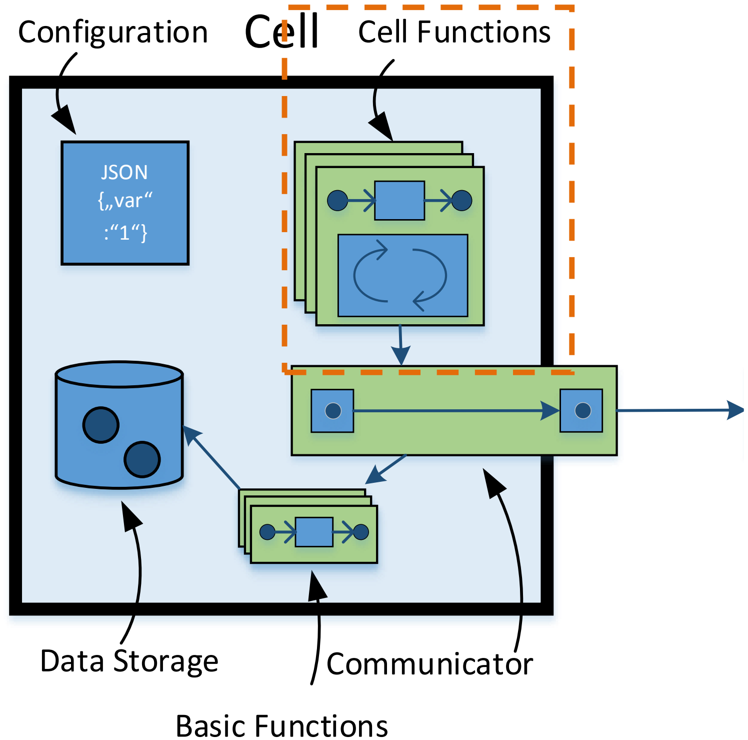
  Serialize class and transfer

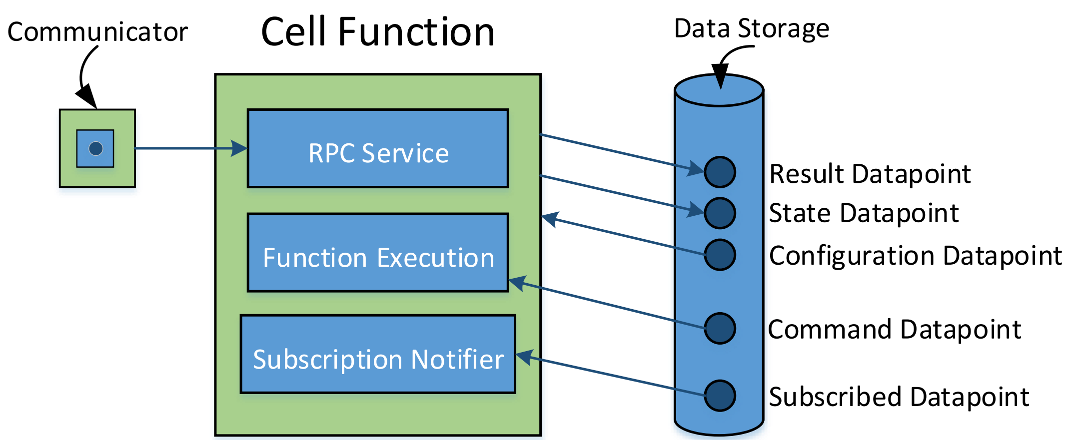
- Function-less data transfer between functions
  - Publisher-subscribe pattern
  - Memory-based data exchange



Cell Function

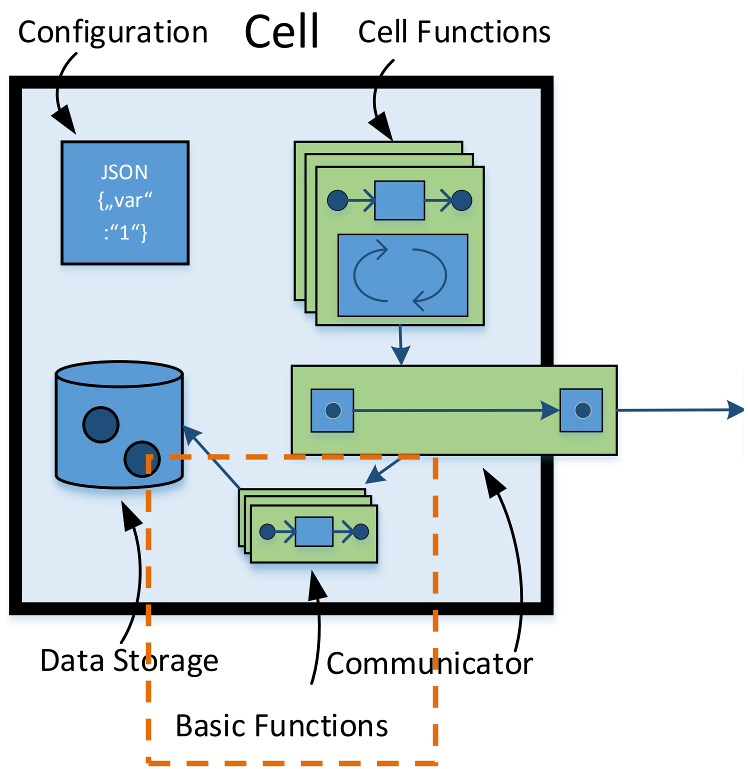
- Defines system functionality
- Multiple activation methods
  - RPC
  - Function execution
  - Trigger on subscription





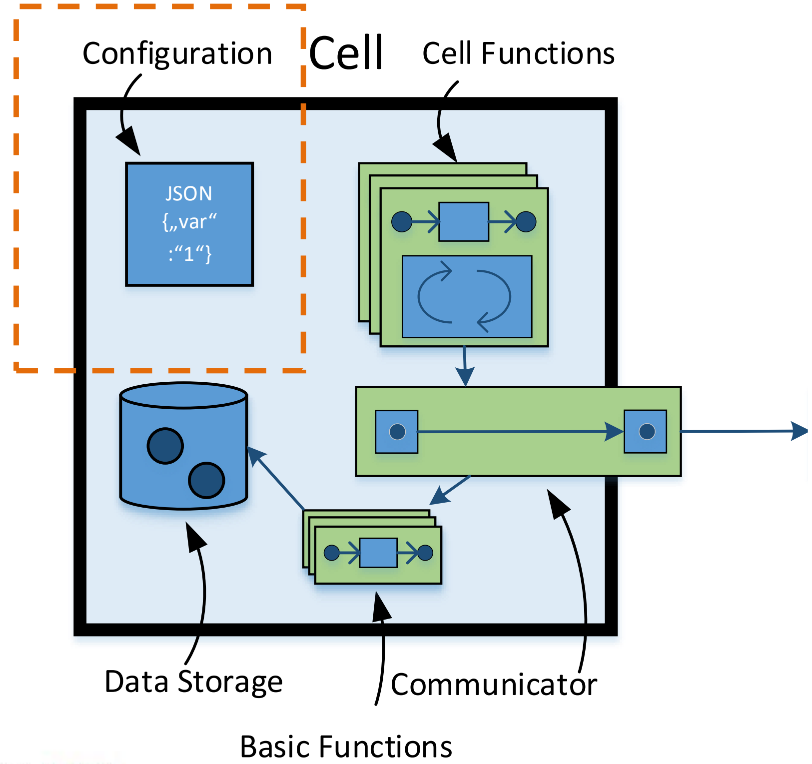
Basic Cell Functions

- Data storage functions
  - Read
  - Write
  - Subscibe
  - Unsubscribe
- Additional functions
  - Collect cell functions states
  - Logger



Configuration

- Used to instantiate the whole cell through reflections



- JSON format

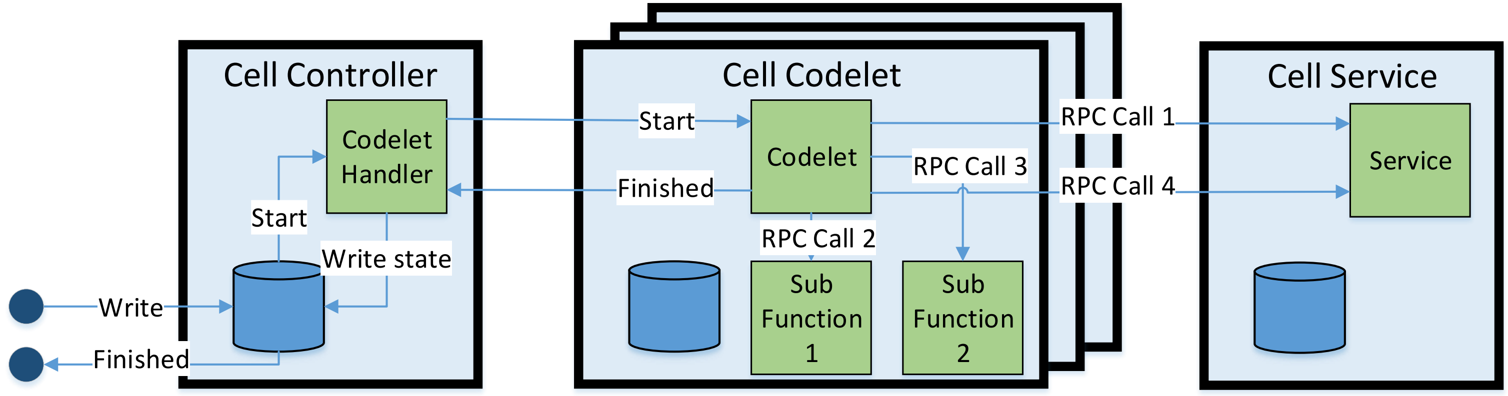
```
CellGatewayImpl weatherAgent3 = this.controller.createAgent(
        CellConfig.newConfig(weatherAgent3Name)
        .addCellfunction(CellFunctionConfig.newConfig(weatherservice, WeatherService.class)
                .setProperty(WeatherService.CITYNAME, "stockholm")
                .setProperty(WeatherService.USERID, "tester")
                .addManagedDatapoint(WeatherService.WEATHERADDRESSID,
                        publishAddress , weatherAgent3Name, SyncMode.WRITEONLY))
        .addCellfunction(CellFunctionConfig.newConfig(CFStateGenerator.class)));

weatherAgent3.getCommunicator().write(DatapointBuilder.newDatapoint(weatherservice + ".command")
        .setValue(ControlCommand.START));
```

Results

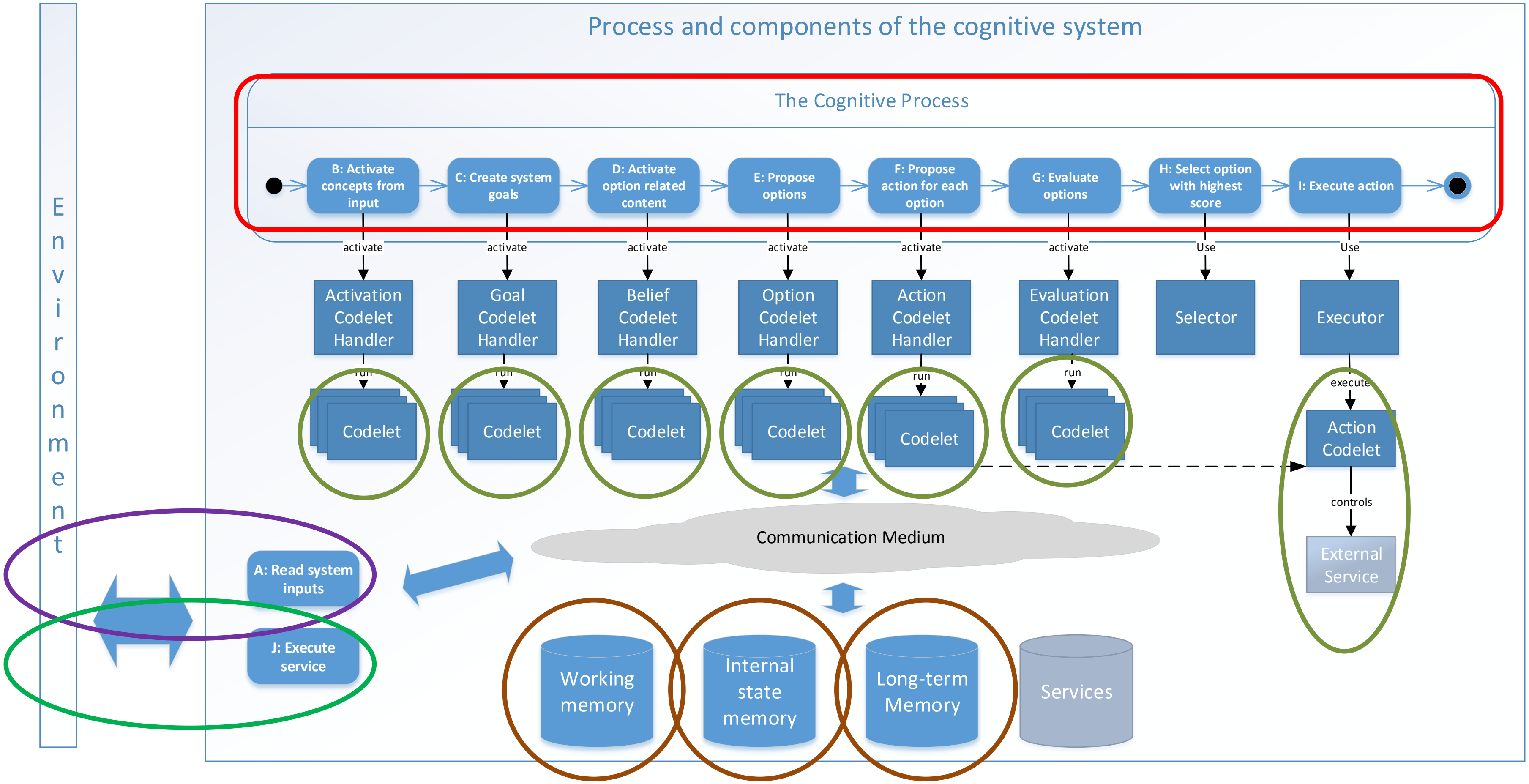
ACONA
https://github.com/aconaframework/acona

- Complex module interactions possible
  - Controller: e.g. web service to receive external calls
  - Codelet/Codelet handler: Rule engine with rules
  - Service: e.g. mathematical algorithm
- Parallel agents and functions with different types of connections



Result: Cognitive Architecture

ACONA
https://github.com/aconaframework/acona



Conclusion and Outlook

ACONA
https://github.com/aconaframework/acona

- Infrastructure extends Java JADE for complex systems
- General and highly customizable
  - Modular system → easy to extend, easy unit testing
  - Functions contain frequent patterns → save developer effort
- Successfully implemented as cognitive architecture

- Next steps
  - Implement cognitive systems in Smart Grid applications
  - Provide framework for evolutionary programming by replicating agents with configuration as “DNA”

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

ICT Energy&IT Group
introducing smart to the electric grid

Institut für Computertechnik
Institute of Computer Technology

Thank You

ACONA

https://github.com/aconaframework/acona

Stefan Wilker

stefan.wilker@tuwien.ac.at

For more information about our projects, please visit
http://energyit.ict.tuwien.ac.at

Performance - Tests

ACONA
https://github.com/aconaframework/acona