ACONA – An Agent-based Architecture for Complex Networks

ICT Institute of Computer Technology

Alexander Wendt
Institute of Computer Technology, TU Wien
Gußhausstraße 27-29 E384,
1040 Vienna,
Austria

alexander.wendt@tuwien.ac.at

ICT Energy&IT Group

introducing smart to the electric grid

Abstract – A large number of research and industrial projects could benefit from a module-based development. However, these modules and the communication between them may vary from project to project. Therefore, a general middleware instead of several specialized middlewares for each domain is desired. This paper presents the ACONA Framework (Agent-based Complex Network Architecture). It is an agent-based middleware with a lightweight and flexible infrastructure. Also, it offers the possibility of evolutionary programming.

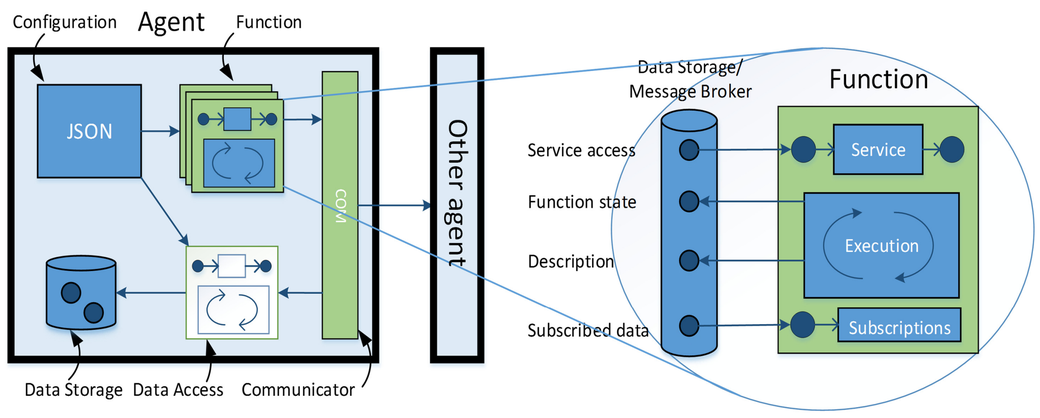
A General Framework

In a heterogeneous system, each system module might be developed in a particular language or by a separate development department. Modules have individual tasks like being device drivers or controllers for applications. Many research and industrial projects show a **demand for module-based development**. Often, it is enough to use a message bus. The developer then implements the custom communication methods tailored for each module. In the case of a typical Smart Grid and IoT application, a message bus for basic communication between sensors and actuators is often enough.

However, in projects that implement multi-agent systems, simulations or cognitive architectures, a bare message bus requires a very high effort of implementing the infrastructure, as they are too simple for the target application. They demands a flexible agent system that offers basic functionality for designing agent functions. The idea is to have one architecture that can cover both demands.

The Agent Model

An agent encapsulates its functions from the environment. Internally, functions share data through a data storage or access offered services. The communication, which is based on Json and MQTT is separated from the function logic. To achieve a general agent, which can replicate, all functions can be generated entirely from a textual configuration.

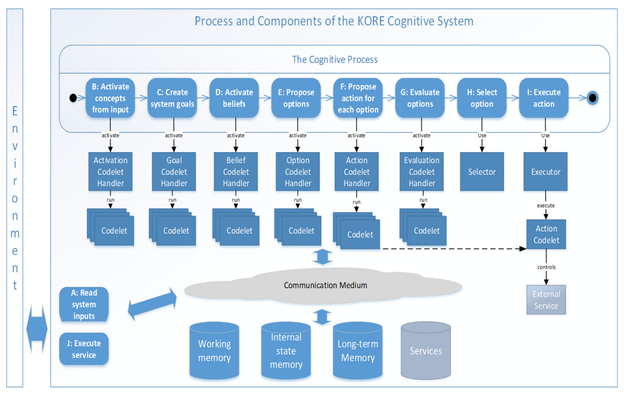


Each function offers the following features:

- Services, which are accessible through other functions (request-response pattern)
- A thread, which runs code on demand or scheduled (parallel execution)
- Listeners, which are triggered by subscribed data points (publish-subscribe pattern)

Services are accessed through a subscribed data point on a message bus.

Artificial Intelligence: Cognitive System

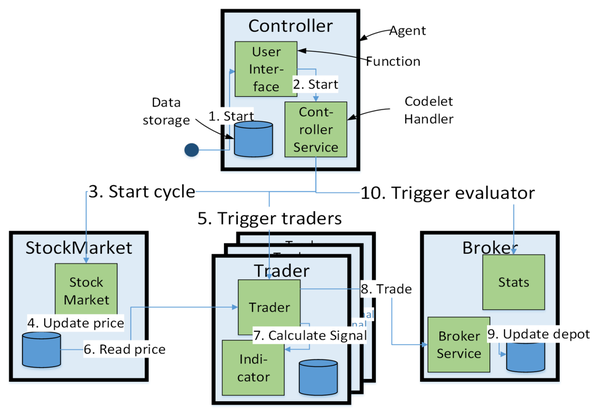


ACONA was used as infrastructure for the modules of a **cognitive architecture** in a project in the area of **building automation**. In a simulation of a building, each room was equipped with temperature, CO2, and occupancy detectors as well as ventilation and heating controllers. Its task was to use an ontology to **generate a set of control strategies** for the building automatically.

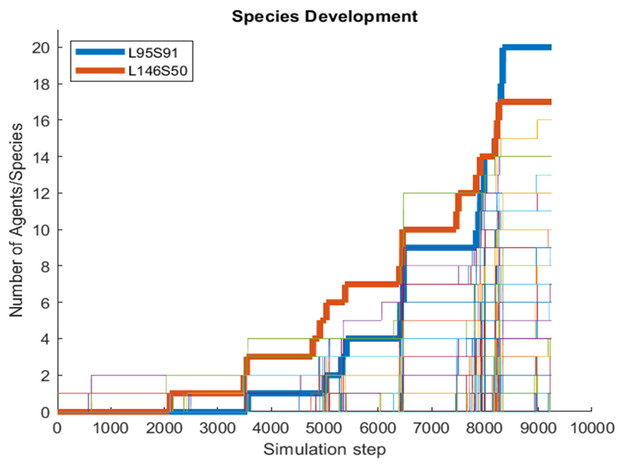
The cognitive architecture was a sequence of steps, each step executed modules in parallel. It consisted of three hierarchical layers. In total ~60 interconnected modules were running. The native data storage of the agent was sufficient to use as different short-term memories.

The system received user requests via a REST interface, loaded data from the Allegrograph triple store, inferred about optimal strategies in an iterative approach and tested these options in a remote simulator.

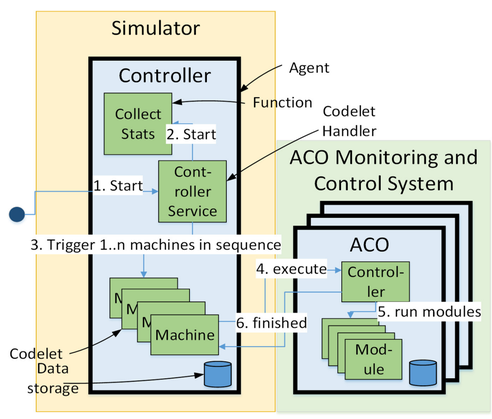
Genetic Programming: Stock Market Simulator



To explore the potential of the framework regarding **self-replication**, **evolution**, and **simulation** capabilities, a stock market trading game was implemented. A **successful trader replicated** itself with a chance of modifications of its parameter. The number of individuals/species measured its success. The system ran successfully with ~8000 agent functions in ~1700 agents.



Multi-Agent-System: Industry 4.0 Application



In an Industry 4.0 project, a **multi-agent system** monitors a number of machines of a conveyor belt. The purpose is to predict and prevent production errors. ACONA was used for **a prototype implementation of both the conveyor belt simulator and the monitoring agents**. The monitoring agents consisted of modules, written in different languages. On one computer the maximum number of running machines+agents was ~800.