

Developing an Agent Systems Reference Architecture

Duc N. Nguyen¹, Robert N. Lass¹, Kyle Usbeck¹,
William M. Mongan¹, Christopher T. Cannon¹,
William C. Regli¹, Israel Mayk², and Todd Urness²

¹ Department of Computer Science
Drexel University, Philadelphia, PA

² Communications-Electronics Research, Development and Engineering Center
U.S. Army, Fort Monmouth, NJ

11th International Workshop on Agent Oriented Software Engineering
10 May 2010

- Problem?** Although standard terms, concepts, and lots of examples exist for agent-based systems, there is no single framework.
- Why?** Without architectural blueprints, it is impossible to understand how the components in an agent-based system should interact, thereby making it difficult to implement and integrate with heterogeneous agent-based systems.
- Solution?** Create an Agent Systems Reference Architecture (ASRA) which provides a template solution for architecting agent-based systems.

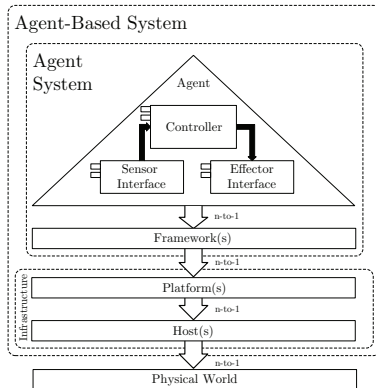
What is an Agent-Based System?

An **agent-based system** consists of many different kinds of agents operating across a heterogeneous set of computing platforms.

An **agent** is an instantiated program within an environment which it can sense and effect.

An agent-based system has four layers:

- 1 **Agent Layer:** the implementation of the application, which achieves the intended functionality;
- 2 **Framework:** the abstraction between agents and the underlying layers which provide agent software functionality;
- 3 **Platform:** a generic infrastructure from which frameworks and agents are constructed; and
- 4 **Environment & Host:** the devices and world in which the infrastructure exists.



What is a Software and Reference Architecture?

A **software architecture** is the structure of the system which contains:

- the software components;
- the external properties of the components; and
- the relationships between the components.

A **reference architecture** is a template solution for a particular domain, which is based on the generalization of successful solutions. A reference architecture contains:

- a list of the software's functions;
- the interaction between internal and external functions; and
- a set of patterns to discuss implementations.

Generalized Technical Approach to Creating an ASRA

There are four steps in our approach:

- 1 Determine the core functions of an agent-based system;
- 2 implement an agent using an existing successful solution which exercises each core function;
- 3 collect data about how the agent executes each function; and
- 4 comprehensively document the design patterns that emerge from the data.

Core Functions: Agent Systems Reference Model (ASRM)

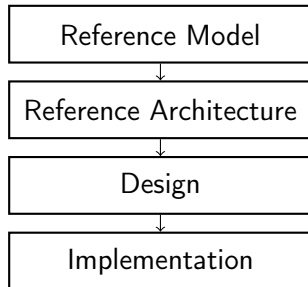
A **reference model** describes the abstract functional elements of a system and does not impose specific design decisions. It provides a common conceptual basis for comparing systems and driving the development of software architectures and other standards.

The **ASRM** defines terminology and concepts to form a reference model for agent-based systems. The ASRM defines seven abstract functional concepts:

- 1 Agent administration;
- 2 security and survivability;
- 3 mobility;
- 4 conflict management;
- 5 messaging;
- 6 logging; and
- 7 directory services.



W.C. Regli, *et al.* Development and Specification of a Reference Model for Agent-Based Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 39(5):572–596, September, 2009.



Implementation: Existing Agent Frameworks

Agent frameworks are the libraries and application programming interfaces (API) for the construction of agents and agent-based systems.

We implement each of the seven function concepts in the following agent frameworks:

- Cougaar;
- JADE; and
- AGLOBE.

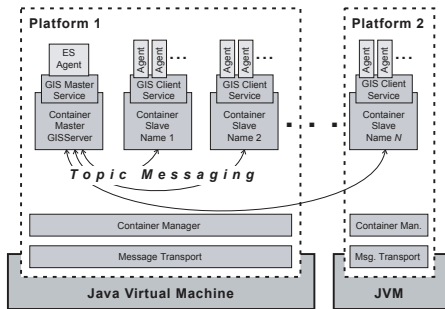


Figure: AGLOBE architecture.

We use reverse engineering techniques to statically and dynamically analyze each agent framework implementing each functional concept.

To accomplish this task, we use two tools:

- 1 the Extensible Java Profiler (EJP) for dynamic analysis; and
- 2 the REportal web-based reverse engineering portal site for static analysis.



S. Mancoridis, *et al.* REportal: A Web-based Portal Site for Reverse Engineering. In *Proc. of the 8th Working Conf. on Reverse Engineering*, pages 221–231, 2001.

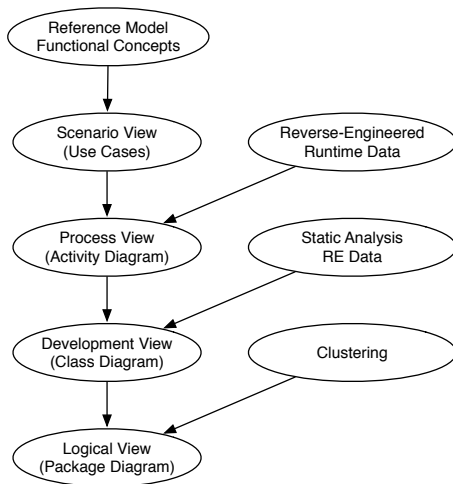
After analysis of the reverse engineering data, we document our findings using the 4+1 View Model. The 4+1 View Model contains five views:

- 1 **Logical View:** the view of the end user (*i.e.*, functionality of the system);
- 2 **Development View (a.k.a Implementation View):** the view of the programmer;
- 3 **Process View:** the view of the dynamic aspects of the system (*i.e.*, the system processes and how they communicate);
- 4 **Physical View:** the view of the system engineer; and
- 5 **Scenarios:** the architecture of the system illustrated through use-case scenarios.

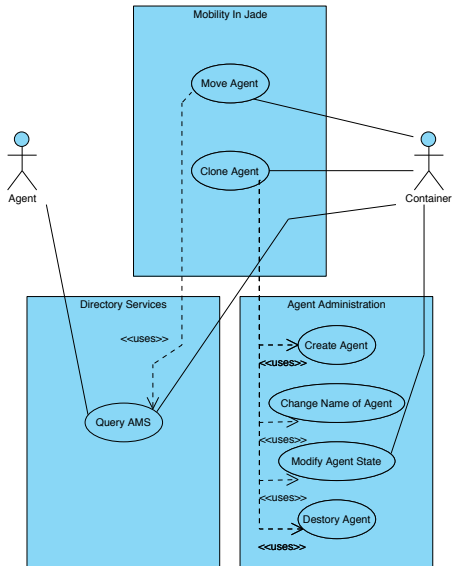


P. Kruchten. Architectural Blueprints—The “4+1” View Model of Software Architecture. *IEEE Software*, 12(6):42–50, November 1995.

Specific Technical Approach to Creating the ASRA

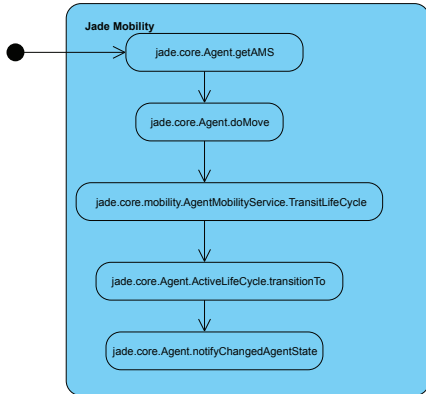


Example: JADE Mobility Scenario View

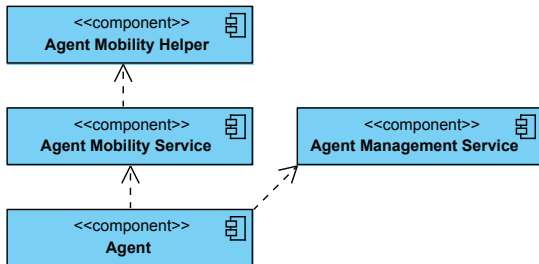


Example: JADE Mobility Process View

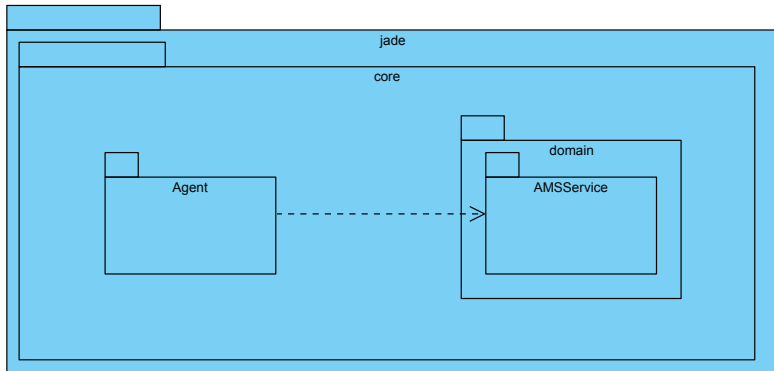
5.4% WhereaboutsAgent\$2.action(...) (5.00 s)
0.0% jade.domain.FIPAAgentManagement.SearchConstraints.<init>(...) (0.00 ns)
0.0% java.lang.Long.<init>(...) (0.00 ns)
0.0% jade.domain.FIPAAgentManagement.SearchConstraints.setMaxResults(...) (0.00 ns)
0.0% jade.domain.FIPAAgentManagement.AMSAgentDescription.<init>(...) (0.00 ns)
3.3% jade.domain.AMSService.search(...) (3.00 s)
2.1% WhereaboutsAgent.moveMyself(...) (2.00 s)
0.0% jade.core.Agent.getAMS(...) (10.00 ms)
0.0% jade.domain.JADEAgentManagement.QueryPlatformLocationsAction.<init>(...) (0.00 ns)
0.0% jade.content.onto.basic.Action.<init>(...) (0.00 ns)
0.5% WhereaboutsAgent.sendRequest(...) (540.00 ms)
0.0% jade.core.Agent.getAMS(...) (0.00 ns)
0.0% jade.lang.acl.MessageTemplate.MatchSender(...) (0.00 ns)
0.0% jade.lang.acl.MessageTemplate.MatchPerformative(...) (0.00 ns)
0.0% jade.lang.acl.MessageTemplate.and(...) (0.00 ns)
0.0% jade.core.Agent.blockingReceive(...) (40.00 ms)
0.0% jade.core.Agent.getContentManager(...) (0.00 ns)
1.0% jade.content.ContentManager.extractContent(...) (1.00 s)
0.0% java.lang.ClassLoader.checkPackageAccess(...) (0.00 ns)
0.0% jade.core.Agent.here(...) (0.00 ns)
0.0% jade.domain.mobility.MobileAgentDescription.<init>(...) (0.00 ns)
0.0% jade.domain.mobility.MobileAgentDescription.setName(...) (0.00 ns)
0.0% jade.domain.mobility.MobileAgentDescription.setDestination(...) (0.00 ns)
0.5% jade.core.Agent.doMove(...) (540.00 ms)
0.5% jade.core.Agent.initMobHelper(...) (530.00 ms)
0.4% jade.core.Agent.getHelper(...) (410.00 ms)
0.1% java.lang.ClassLoader.loadClassInternal(...) (120.00 ms)
0.0% jade.core.Agent\$1.<init>(...) (0.00 ns)
0.0% jade.core.mobility.AgentMobilityService\$AgentMobilityHelperImpl.registerMovable(...) (0.00 ns)
0.0% jade.core.mobility.AgentMobilityService\$AgentMobilityHelperImpl.move(...) (10.00 ms)
0.0% jade.core.mobility.AgentMobilityService\$TransitLifeCycle.<init>(...) (0.00 ns)
0.0% jade.core.mobility.AgentMobilityService\$TransitLifeCycle.<init>(...) (0.00 ns)
0.0% jade.core.LifeCycle.<init>(...) (0.00 ns)
0.0% jade.core.mobility.AgentMobilityService.getName(...) (0.00 ns)
0.0% jade.util.Logger.getLogger(...) (0.00 ns)
0.0% jade.core.Agent.changeStateTo(...) (10.00 ms)
0.0% jade.core.LifeCycle.setAgent(...) (0.00 ns)
0.0% jade.core.LifeCycle.equals(...) (0.00 ns)
0.0% jade.core.Agent\$ActiveLifeCycle.transitionTo(...) (0.00 ns)
0.0% jade.core.Agent.notifyChangedAgentState(...) (10.00 ms)



Example: JADE Mobility Development View



Example: JADE Mobility Logical View



Due to page limitations, this paper only provides an overview of the ASRA. The complete ASRA:

- Provides a template solution for an architecture for agent systems;
- maps the terms defined in the ASRM to architectural blueprints;
- describes best practices for defining architectural paradigms to turn blueprints into architectures; and
- identifies a set of design patterns to build architectural components.
- Document the seven functional concepts for each agent framework (e.g., Cougaar, JADE, and AGLOBE) using the 4+1 View Model; and
- identify design patterns for each functional concept.

The contributions of this work are:

- ① A methodology for creating an ASRA through reverse engineering and the 4+1 View Model;
- ② a mapping of the ASRM's functional concepts to architectural blueprints; and
- ③ design patterns to build concrete architectural components.

Future work consists of creating design paradigms for:

- Agents and external systems outside traditional agent-based environments; and
- agent societies and communities.

Thank you for your time and attention.

Duc N. Nguyen

`dn53@cs.drexel.edu`

<http://cs.drexel.edu/~dn53/>

Robert N. Lass

`urlass@cs.drexel.edu`

<http://cs.drexel.edu/~urlass/>

William M. Mongan

`wmm24@cs.drexel.edu`

<http://cs.drexel.edu/~wmm24/>

Kyle Usbeck

`kfu22@cs.drexel.edu`

<http://cs.drexel.edu/~kfu22/>

Christopher T. Cannon

`ctc82@cs.drexel.edu`

<http://cs.drexel.edu/~ctc82/>

William C. Regli

`regli@cs.drexel.edu`

<http://cs.drexel.edu/~regli/>