

Build Process

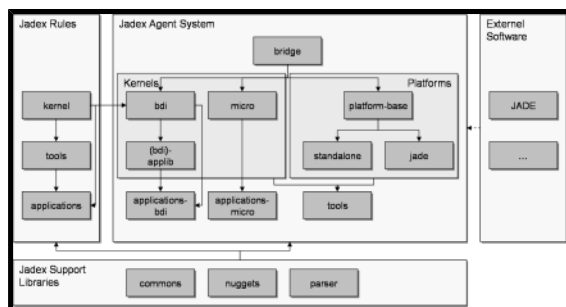
The details of the build process are very different with respect to the different Jadex versions. In the following the steps for Jadex V2 are explained.

Building Jadex V2

Basically Jadex V2 is modular project, which consists of many submodules with clearly defined dependencies. The overall build process is defined using the [Maven](#) tool.

Project Structure

In this section the basic Jadex project structure is briefly explained. It is very helpful to know what the basic functionalities of the different modules are and which dependencies between them exist in order to understand how Jadex V2 works and at which spots extensions could be possibly added.



Jadex packages

On an abstract level Jadex is composed of four main building blocks:

Jadex Rules

Jadex Rules is a general purpose forward-chaining rule system. It is comparable to other rule engines like [JESS](#) and [Drools](#), but has been developed with a special focus on a small and lightweight design, which facilitates the usage in agents. This package is currently used as basis for the BDI kernel of the Jadex agent system. Relying on a general purpose rule engine allows for efficient rule execution and simplifies the optimization of the BDI kernel. Jadex Rules is a self-contained project on its own (cf. [Jadex Rules Home](#)).

Jadex Support Libraries

The Support Libraries contain small helper functionalities of general applicability. The 'commons' packages mainly contains static helper classes for e.g. enhanced reflection functionalities. In addition, the 'nuggets' package is a highly efficient general purpose XML to Java and vice versa converter. In the Jadex agent system nuggets is mainly used for marshalling and unmarshalling message content. Finally, the 'parser' package is a (slightly extended) Java parser. It allows for parsing arbitrary complex Java (1.4) expressions (right hand side of an assignment). The parsed expressions can be directly evaluated (possibly with parameters) to Java objects. The parser is utilized in the Jadex agent system for evaluating Java expressions from XML description files of agents (e.g. for belief values).

External Software

External libraries are used where appropriate. One specific aspect here is the construction of agent platforms relying of existing middleware. E.g. the JADE adapter of the Jadex agent system wraps the well-known [JADE agent platform](#) and reuses its services for controlling Jadex agents.

Jadex Agent System

The Jadex agent system is based on the explicit distinction between kernels and platforms. A kernel is the realization of a specific agent architecture such as BDI (belief-desire-intention), rule-based or task-based, whereas a platform is responsible for executing agents and providing the necessary management functionalities e.g. for starting/stopping agents and registering/searching for services. To avoid a direct dependency between kernels and platforms the common base package 'bridge' has been devised. It contains functionalities that are needed by kernels as well as platforms and mainly consists of interfaces e.g. for an agent identifier.

On the kernel side, currently two different implementations are available. The BDI kernel allows for programming intelligent agent based on mentalistic (human-like) notions like beliefs and goals. An extensive library (applib) is available for this kernel. It provides reusable functionalities e.g. for interaction protocols like contract-net and interaction with management services like AMS and DF. In contrast, the micro kernel is a minimal agent kernel implementation, which only offers very basic functionalities for the agent programmer. Nonetheless, these agents are extremely small and efficient and can e.g. be used for conducting simulation experiments with a high number of agents (over 100.000 in a standard JVM with 64MB memory are possible). All agents rely on platform services for communication means. Hence, agents of different kernels can easily communicate and also hybrid systems (using BDI agents for complex tasks and micro agents for simpler ones) are possible.

On the platform side, currently the Standalone and the JADE implementations are available. The standalone platform is a lightweight, extensible, highly-efficient realization for running Jadex agents. It is based on the service metaphor and provides all its functionalities via so called platform services. These services can be easily configured using a configuration file (using a custom property or a [Spring](#) beans descriptor). With this approach the Standalone platform can be easily adapted to different use cases, e.g. a streamlined version for mobile devices can be built. As all aspects of the platform can be customized using services the Standalone platform can also be configured as simulation infrastructure providing event-driven and time-driven execution modes. The JADE adapter reuses the JADE agent platform as middleware for executing agents and thus allows a FIPA-compliant communication with other agent platforms.

Finally, packages for tools as well as example applications exist. One aim is that the tools should be independent from the concrete kernel and hence be reusable to a great extent. As this is not achievable for all kinds of tools (e.g. a BDI introspector), in future versions a separation to general purpose and kernel specific tools will be provided. The tools are integrated in the so called Jadex Control Center (JCC), which represents the central access point for all further tools. These tools are developed as plugins, so that the JCC can be configured in a flexible way and new tools can be added without much efforts.

Building with eclipse

- Download [Jadex V2](#) from sourceforge and unzip it to some directory
- If not present on your system install a new Java SE Development Kit (JDK) version from [Sun](#)
- If not present on your system install [eclipse version 3.4.1](#) or newer
- Start eclipse and select some workspace

Install the m2eclipse plugin

- In menu Help --> Software Updates use tab "Available Software" and click on "Add Site..."
- Enter the plugin URL <http://m2eclipse.sonatype.org/update/> (Please note that the development version of maven currently does not work with the existing poms as the transitive dependency policies seem to have changed)
- Select all checkboxes except 2.4 "Maven SCM handler for Subclipse (Optional)" and 3.1 "Maven Integration for AJDT (Optional)"

Import Maven project

- Use File --> Import --> General --> Maven Projects
- Switch to the Jadex directory (where you unzipped the distribution)
- Select the .pom file from that directory

Resources - Build Process

- Maven will start the build process and download the necessary libraries from the web
- Due to a bug in the m2e plugin after the build errors in the modules "jadex-applications-bdi" and "jadex-bridge" might appear
- Open these modules and change the used JRE by rightclicking on "JRE System Library..." and selecting the properties
- In the properties dialog select "Workspace default JRE"

Launching

- There are several predefined launch configuration available. Go to the folder src --> main --> config in module "jadex-launch"
- Rightclick on the file "Jadex Standalone.launch" and select "Run as --> Jadex Standalone"