

Features

The following sections highlight some of the current features of Jadex. In summary, Jadex is an **agent-** and **Java-based, modular, conceptually leading, technology affine** and **standards compliant**, agent environment, and allows to develop **goal oriented agents** following the BDI model. Jadex provides a **framework** approach including a runtime infrastructure for agents, the agent platform, and an extensive runtime tool suite.

Agent-based

The agent concept is regarded as a powerful software development paradigm, well suited to address the complexity of today's large software systems. It can be considered as a natural extension to object-orientation, whereby an agent represents an object with control about its execution. The agent paradigm allows to view a system as being composed of autonomous interacting entities which pursue their own goals and act in a rational manner. Interestingly, the agent metaphor fits perfectly to the demands of **distributed and concurrent systems**. In a multi-agent system each agent is autonomous from other agents and hence potentially concurrently executed to the others. In addition, agents communicate via messages and do not need a shared memory environment (like objects using method calls).

Java-based

The Jadex project aims to make the development of agent based systems as easy as possible without sacrificing the expressive power of the agent paradigm. To foster a smooth transition from traditional distributed systems to the development of multi-agent systems, well established **object-oriented concepts** and technologies should be employed wherever possible. With Jadex you are able to create agent systems without having to learn a new programming language. Jadex is designed to facilitate the implementation of agents in the widespread Java programming language, therefore allowing to reuse a vast amount of existing tools and libraries.

Modular Design

Jadex has been developed in a very modular fashion. This modular design has one major goal: the clean separation between the agent platform and the agent kernel. This separation allows a many-to-many relationship between agent platforms (i.e. Standalone or JADE) and kernels (i.e. BDI or Micro). I.e., on the one hand, on a specific agent platform different kernels can be run (also concurrently) and hybrid agent applications can be built. On the other hand, a specific agent kernel can also be run on different agent platforms (requirement is that a Jadex adapter has been built). This allows the execution of e.g. BDI agent programs on the Standalone or JADE platform without requiring any changes to the agent code.

Conceptually Leading

The Jadex framework is developed since 2003. From the beginning, the project has the aim to reveal and provide the best possible support for building distributed and concurrent programs. This class of software gains ever increasing importance by many factors and current trends such as ubiquitous computing and multicore processors. Hence, Jadex has always been driven by new research ideas and incorporates these as new features as soon as possible in the software. Some examples of these outstanding features are:

- Incorporated event- and time-driven simulation support
- Multi agent kernel support
- Full BDI-reasoning cycle support (goal deliberation and means-end reasoning)
- Goal-oriented interaction protocols
- Many prebuilt functionalities readily available in capabilities

Technology Affine

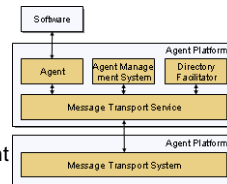
About - Features

Even though Jadex is driven by research ideas, the technology side is considered equally important and non-functional aspects like robustness, scalability, fault tolerance etc. have been targeted specifically in the new Jadex V2 design. It is always tried to exploit the best of breed software for Jadex and make use of proven technology. This is reflected e.g. by:

- Maven is used as build and release tool
- The Standalone platform realizes a services based approach and can be configured e.g. with Spring Beans
- BDI kernel is built on established and efficient rule technology (RETE engine)

Standards Compliant

Standards are important to ensure that agreed best practices are used and standards compliance allows consumers of technologies to exchange products of different vendors lowering the risks of a vendor lock-in. In the field of multi-agent systems only a few standards have been developed so far. Jadex aims at following standards wherever possible. This is not restricted to agent standards. Most important in the field of agent standards are the communication standards, which should enable interoperability between different agent platforms. These standards have been released by the [Foundation for Intelligent Physical Agents \(FIPA\)](#), which are commonly referred to as "the FIPA standard". As shown in the figure, the FIPA standard specifies an agent platform architecture, which defines services such as agent management and a directory facilitator. This architecture enables agents to communicate using a common agent communication language. To achieve FIPA-compliance, Jadex can be used with the [JADE Agent Framework](#), an open source development by the [Telecom Italia Lab](#). JADE provides the platform architecture and the core services and message transport mechanisms as required by the FIPA specifications.



Goal Oriented Agents

The internal state and decision process of agents is therefore modelled in an intuitive manner following the notion of **mental attitudes**. Goal orientation means that, instead of directly requesting the agents to perform certain actions, the developer can define more abstract goals for the agents, thereby providing a certain degree of flexibility on how to achieve the goals. The BDI Model, based on the mental attitudes belief, desire and intention, was first introduced as a philosophical model for modelling rational (human) agents, but later adopted and transformed into an execution model for software agents, based on the notion of beliefs, goals, and plans. Jadex incorporates this model by introducing **beliefs**, **goals** and **plans** as first class concepts that can be created and manipulated inside the agent. In Jadex, agents have beliefs, which can be any kind of Java object and are stored in a belief base. Goals come in different flavors and allow to describe what an agent should achieve in a declarative manner. Goals do not include information about how it can be fulfilled. This kind of knowledge is encoded in plans. Hence, to achieve its goals the agent executes plans, which are procedural recipes coded in Java.

Framework

Programming Jadex agents can be done by using the framework **API** and possibly predefined **reusable generic functionality** in form of capabilities. The API provides access to the Jadex concepts when programming plans. Plans are plain Java classes, extending a specific abstract class, which provides useful methods e.g. for sending messages, dispatching sub goals or waiting for events. Plans able to read and alter the beliefs of the agent using the API of the belief base. A special feature of Jadex is that in addition to directly retrieving stored facts, an intuitive **OQL-like query language** allows to formulate arbitrary complex expressions using the objects contained in the belief base. In addition to the plans coded in Java, the developer provides an *XML based Agent Definition File (ADF)*, which specifies the initial beliefs, goals, and plans of an agent. The Jadex runtime engine reads this file to instantiate an agent model, and executes the agent by keeping track of its goals while continuously selecting and executing plan steps, based on internal events and messages from other agents. Jadex is supplied with some predefined functionality e.g. to access a directory facilitator service. The functionality, coded in

About - Features

separate plans, is composed in reusable agent modules called **capabilities**, described in a format similar to the ADF, and can easily be plugged into existing agents.

An important quality aspect of any development environment is the available tool support. As Jadex uses the standard programming languages XML and Java existing IDEs like eclipse can be used for programming agents as well. Furthermore, Jadex offers a complete runtime tool suite for managing, testing and debugging agents. Basis of this suite is the Jadex Control Center tool (JCC), which is the access point for all the tools (realized as plugins). Currently the following tools are available:

- **Starter** for starting and stopping agents and viewing all agents currently living on the platform
- **Directory Facilitator** for viewing registered services of agents
- **Conversation Center** for directly sending messages to agents
- **Comanalyzer** for sniffing and displaying message-based interactions
- **Test Center** for executing agent-based unit tests
- **Simulation Control Center** for executing applications in real-time or simulation modes (time- or event-driven)
- **Introspector** for introspecting agents and executing them stepwise
- **Library Tool** for managing the Java classpath at runtime by adding/removing libraries such as jar-files