Agent Toolkits for Ad Hoc Grids

Tariq Abdullah, Koen Bertels

Computer Engineering Laboratory, EEMCS, Delft University of Technology,
Mekelweg 4, 2624 CD, Delft, The Netherlands
{m.t.abdullah, k.l.m.bertels}@tudelft.nl

Abstract. Autonomic computing systems can manage themselves by self-configuration, self-healing, self-optimization, and self-protection. Software agents are the promising candidates for making the autonomic systems a reality due to their characteristics. Different research projects use software agents for developing autonomic computing applications. Agent development toolkits address different aspects of the software agents. In this paper, firstly, we present a categorization of the agent development toolkits from different aspects of autonomic computing paradigm. Secondly, recommendations for selecting an appropriate toolkit while developing an autonomic ad hoc grid will be given. Finally, results of micro economic based resource discovery in a local ad hoc grid are presented.

1 Introduction

There has been an explosive growth in computation, communication, data storage and integration technologies. At present, there are millions of commercial and/or research organizations, and trillions of computing devices. All these organizations establish some form of LAN/WAN for their computational requirements. Skilled staff are required to maintain these computational infrastructures and this requirement increases with an increase in the complexity of these systems.

Ad hoc grids are dynamic, heterogeneous and have complex infrastructure. The complexity for running and managing the ad hoc grids is more higher. Agents with autonomy, mobility, learning, intelligence, reactivity, social-ability, and pro-activeness are promising candidates for making the autonomic systems a reality. Different projects [1,2,3] used agents for developing autonomic computing based applications. Agent development toolkits are needed to develop software agents and agent based applications. Some studies attempted to compare agent development toolkits, as discussed in Section 3, with focus on different aspects of the agent development toolkits.
In this paper, firstly, we discuss how software agents can be used for developing autonomic computing applications. Secondly, we categorize the agent development toolkits from different aspects of autonomic computing paradigm. Thirdly, recommendations for selecting an agent development toolkit while developing an autonomic computing application are given. Finally we present results from our ad hoc grid middleware, which is developed by using the selected agent development toolkit.

The rest of the paper is organized as follows. Section 2 gives an overview of autonomic computing and describes how different properties of the software agents can be used for different aspects of autonomic computing. Section 3 describes agent development toolkits and also describes the agent properties supported by each toolkit. This section also gives some recommendations for developing autonomic ad hoc grids by using the surveyed agent development toolkits. The experimental setup and results are presented in Sections 4 & 5 respectively. While, Section 6 concludes the paper and describes future research directions.

2 Autonomic Computing & Software Agents

Autonomic Computing (AC) was coined by Paul Horn in 2001, from IBM, when he suggested to, "build computer systems that regulate themselves much in the same way as our autonomic nervous system" [4]. Since then the research community has attempted to define AC, its properties, evaluation criteria and its integration with other branches of knowledge. There is no unique definition of AC. It can be defined with the help of its basic properties listed below:

- **Self Configuration:** This is the ability of a system to automatically and effectively configure and reconfigure itself under varying and even unpredictable conditions.
- **Self Healing:** This is the ability of the system to recover from routine and extra ordinary events. The system must be capable of finding potential problems by itself and finding alternate ways of using available resources to reconfigure and thus keep itself functioning smoothly.
- **Self Optimization:** This is the ability of an AC system to search for ways of optimizing its operations by monitoring its constituent parts and by fine tuning them to achieve system goals.
- **Self Protection:** This is the ability of a system to identify attacks.

Software agents possess certain properties that can be related to AC system properties. First, we describe the agent properties, and then the properties will be associated with different aspects of AC systems.
Every agent has a *unique identity* within a well-defined boundary and interfaces. Every agent has particular *design objectives* that are represented implicitly or explicitly. An agent is *autonomous* when it has control over its internal state and its behavior. Every agent needs to be *reactive* by timely responding to changes that occur in its environment in order to satisfy its design objectives. An agent also needs to adopt new goals and take initiative in order to satisfy its design objectives by being *proactive*. An agent can be *mobile*. An agent exhibits weak mobility when it can only migrate its data from one environment to some other environment and exhibit strong mobility when it can migrate processes as well as data. An agent can exhibit *social-ability* by communicating with other agents. An agent can *learn* knowledge from its environment as well as from its past experiences. When an agent applies its knowledge according to its circumstances in order to fulfill its design objectives, it is called *an intelligent agent*. An agent

AC Characteristics	Agent Characteristics
Self Configuration	autonomous, mobile, learning
Self Optimization	autonomous, intelligent, learning, reactive
Self Healing	intelligent, reactive, mobile, autonomous
Self Protection	proactive, reactive, intelligent, secure, mobile

Table 1: Agent Characteristics for Autonomic Computing

needs to be *interoperable* so that it can communicate and exist in different operating environments.

Agents with autonomy, mobility, learning, reactivity, social-ability, intelligence and pro-activeness are promising candidates for making the autonomic systems a reality. A number of research projects attempted using software agents for developing autonomic computing based applications.
Components of AC can be expressed in terms of agent properties. Autonomous, learning and mobile agents are needed for *self-configuration.* Intelligent, reactive, learning and autonomous agents are needed for *self-optimization.* Intelligent, autonomous, reactive, and mobile agents are needed for *self-healing*, whereas autonomous, proactive, reactive, intelligent, secure and mobile agents are needed for *self-protection.* Table 1 provides an overview of agent characteristics suitable for different aspects of AC.

3 Agent Development Toolkits

There are several agent development toolkits projects. Some projects are completed and obsolete while some are continuously evolving. There has been efforts in the research community for comparing different agent development toolkits with each other and for their suitability in different applications. For example, performance evaluation on message transport system in JADE [5], Zeus [6] and in JACK [7] is focused in [8]. Shakshuki et al. [9] compared Aglets, Voyager, Odyssey, and Concordia for agent mobility. Vbra et al. [10] compared JADE, FIPA-OS, ZEUS, and JACK. This comparison focused on interoperability, cost, security and memory footprints of these Java based platforms. Chmiel et al. [11] analyzed agent creation, message exchange, database excess, inter/intra container communication for JADE.

All the above mentioned studies only compared some aspects of the software agents in the agent toolkits. However, these only compared one or two toolkits with each other. In this section those research (open source) / commercial agent development toolkits are described in detail, which are in continuous development process and support some of the features of autonomic computing. A summary of these toolkits is given in Table-2.

1. Agent Factory (AF) [12] is a java based, FIPA (Federation of Intelligent and Physical Agents) [13] complaint, open-source cohesive framework that

supports a structured approach to the development and deployment of agent-based systems. It has a layered architecture. It includes an agent oriented programming language for development known as AF-APL (Agent Factory-Agent-oriented Programming Language). Each AF agent has its own thread of control and communicates through Agent Communication Language (ACL). The AF toolkit is used in developing ubiquitous computing, robotics, enterprise search and distributed sensor network applications.

2. Ajanta [14] is a Java based, non-FIPA compliant, mobile agent programming system. It supports secure mobile agents and agent based applications over the Internet. Ajanta supports strong agent mobility. Ajanta mobile agents travel autonomously from machine to machine on a network to achieve their design objectives, and communicate via RMI. The Ajanta toolkit is the only toolkit available for Unix-like platforms and has been used to develop global file access system and applications for distributed collaboration.
3. Agent Development Kit (ADK) [15] is a Java based, FIPA compliant, agent platform with emphasize on secure mobile agents. An ADK agent is an autonomous piece of code that can carry data and travel the network to perform a task for its owner and uses HTTP, HTTP(s) and JMS for agent communication. It supports strong agent mobility, follows task oriented programming model, where tasks are arranged like work-flows. It can run on a variety of platforms, ranging from IBM zSeries to mobile phones. ADK can be integrated with J2EE server and is distributed under the LGPL and a proprietary license. It has been mainly used in several commercial projects especially for the application integration of legacy systems. DataExplorer and FROG are some examples that use ADK.
4. AgentBuilder (AB) [16] is a commercial, Java based, non-FIPA compliant agent platform to develop intelligent agents and agent based applications. It is an integrated toolkit that supports all phases of agent software development. It supports KQML, CORBA and TCP/IP for agent communication and is available in two different versions, namely, AgentBuilder LITE and AgentBuilder PRO. AgentBuilder LITE is ideal for developing single-agent, stand-alone applications and small agencies, while AgentBuilder PRO has all features of AgentBuilder LITE and an advanced suite of tools for testing and developing intelligent agents and multi-agent systems. It can run on Solaris, Windows, and Unix/Linux. Applications like mail, auctions and shopping agents have been developed using AgentBuilder.
5. Decentralized Information Ecosystem Technologies (DIET) [17] was developed as part of EU project. It aims at developing scalable, lightweight, and robust agent platforms for P2P and/or adaptive distributed applications. Agents developed in DIET are autonomous and lightweight. Communication between local agents is via message passing. Projects like DIANE and SWAN used DIET in their development. The project was completed in 2003 and since then it has been declared an open source project.
6. JACK [7] is a Java based, non-FIPA compliant, commercial agent development toolkit for developing intelligent agents based on BDI model. JACK incorporates a set of tools like “design tool”, “plan editor”, and “graphical plan

tracing". Agents communicate through TCP. It can run on Solaris, Windows, Unix/Linux and Mac OS. JACK supports CORBA, RMI, J2EE, EJB, .NET, or DCOM for external package integration. It is used in developing systems for unmanned aerial vehicles and human-like decision making.

7. ZEUS [6] is a Java based, open source, FIPA Compliant, component oriented agent development toolkit. It provides facilities to implement BDI style agents with reactive rule bases and with intelligent message handling. The ZEUS agents use FIPA ACL for communication.
8. Java Agent DEvelopment Framework (JADE) [5,18,19] is a Java based, FIPA-compliant, open source agent development framework. It simplifies the implementation of a Multi Agent System (MAS) through a middleware and a set of graphical tools. The basic agent functions and behaviors are provided through its API. The JADE agents communicate through TCP/IP, and ACL. The JADE agents can be executed on Unix, Windows or on both platforms. JADE supports weak agent mobility. JADE also support web services integration and can integrate, JAVA implementation of, JESS reasoning engine to make JADE agents intelligent and reactive. JADE has been used to develop a number of commercial and non-commercial projects. The projects like MAST at Rockwell automation, BT Exact, Whitestein Technologies, Singular software, Acklin B.V., Knowledge on Demand, CoMMa, TeSCHeT and a number of different universities are using JADE for research projects. We are using JADE in GRAPPA project[20].
   Some commonly used extensions of JADE are as follows: **JADE-LEAP** (Java Agent DEvelopment Framework-Lightweight Extensible Agent Platform) [21] which enables developers to develop and execute FIPA-compliant multi-agent systems in mobile devices like cell phones and palm computers. **BlueJADE** is JADE extension for J2EE applications under JBoss. **JADEX** (JADE extension) [22] is an agent layer on top of JADE that allows easy development of rational, intelligent agents with mental attitudes by following the BDI model.

Name	**Organization / University**	**Language**	**FIPA**	**Communication**	**License**	**Mobility**
ADK	Tryllian	Java	yes	HTTP(s), JMS	Dual	yes
Agent Builder	Acronymics	Java	no	KQML, TCP/IP	Commercial	no
Agent Factory	UC Dublin	AF-APL2	yes	FIPA ACL	Open source	no
Ajanta	Uni. of Minnesota	Java	no	RMI	Open source	yes
DIET	Consortium	Java	no	Message Passing	Open Source	no
Jack	AO Software	Java	yes	TCP	Commercial	no
JADE	Tilab	Java	yes	TCP/IP	Open Source	yes
ZEUS	BT	Java	yes	TCP/IP	Open Source	no

Table 2: Agent Development Toolkits

This section describes the support for different agent properties provided by different agent development toolkits. We now give some recommendations about the suitability of different agent development toolkits for autonomic computing. It can be concluded that the toolkits supporting autonomy, interoperability, mobility, intelligence, reactivity, pro-activeness and social ability of agents are suitable for most aspects of AC based systems. AgentBuilder, JACK, ZEUS and JADE fall in this category. Toolkits supporting autonomy and social ability of agents are not suitable for autonomic computing based systems. DIET and AgentFactory fall in this category. Toolkits supporting autonomy, interoperability, mobility and social ability of agents are partially suitable for Self-configuration and Self-healing of AC based systems. ADK and Ajanta fall in this category. Self-configuration is partially/fully supported by all the surveyed toolkits. Whereas, self-healing is supported by all the surveyed toolkits, except AgentFactory and DIET. Self-optimization and self-protection are partially/fully supported by AgentBuilder, JACK, JADE and ZEUS. JACK and ZEUS support most aspects partially. However JADE fully supports all aspects of AC.

	Autonomous	Mobile	Interoperable	Learning	Intelligent	Secure	Reactive	Proactive	Lightweight	Social ability
ADK	y	y	y	n	n	y	n	n	y	y
Ajanta	y	y	n	n	n	n	n	n	n	y
AB	y	n	y	y	y	n	y	y	n	y
JACK	y	n	y	y	y	n	y	y	n	y
JADE	y	y	y	y	y	y	y	y	y	y
AF	y	n	y	n	n	n	n	n	n	y
ZEUS	y	n	y	n	n	n	y	y	n	y
DIET	y	n	n	n	n	n	n	n	y	y

Table 3: Agent Development Toolkits Supporting Different Agent Characteristics

	ADK	AB	AF	Ajanta	DIET	JACK	JADE	ZEUS
Self Configuration	Partially	Partially	Partially	Partially	Partially	Partially	Yes	Partially
Self Healing	Partially	Partially	No	Partially	No	Partially	Yes	Partially
Self Optimization	No	Yes	No	No	No	Yes	Yes	Yes
Self Protection	No	Partially	No	No	No	Partially	Yes	Partially

Table 4: Supported Aspects of AC by Agent Toolkits

These conclusions are summarized in Tables 3 & 4. Table 3 summarizes the discussed toolkits and their support for different agent properties and Table 4 summarizes agent toolkits supporting different aspects of autonomic computing. In the end, we mention that these recommendations are based on the present feature set of the surveyed toolkits. As majority of these toolkits are in continuous development, it is recommended to check the latest feature set of any toolkits before choosing any specific toolkit.

In the next sections, we present the experimental setup and discuss the results of microeconomic based resource allocation approach, developed on top of JADE, for a local ad hoc grid.

4 Experimental Setup

An overview of the system components and experimental setup of the proof of concept application in JADE is described in this section. The proof of concept application is an agent based middleware for resource allocation in a local ad hoc grid. The overall architecture of the middleware is represented in Figure 1. The *Index* agent (un)registers all node agents. The *SQL* agent is responsible for database operations. The *Node* agent consists of *consumer* and *producer* agent. Each node agent will behave as a consumer or as a producer of resources at any given time. The consumer agent is responsible for generating a resource request and submitting the request to the producer agent. The producer agent generates a resource offer, receives resource requests from consumer agent, initiates auctioneer agent and executes the consumer job. The *Auctioneer Agent* performs matchmaking by using Continuous Double Auction (CDA). The details of CDA and ask/bid prices can be found in [23]. The *communication agent* handles all communication between different agents by using TCP/IP and FIPA ACL.
The system works as follows. Each node agent registers with the index agent while joining the ad hoc grid. The consumer agent retrieves the available producer agents from the index agent. The consumer agent prepares and sends a *resource request* to all available producer agents. The producer agent receives resource request(s) from consumer agents during the bidding period. The producer agent performs matchmaking for its available resources from the received resource requests. The auctioneer agent of the producer agent performs matchmaking using Continuous Double Auction (CDA). The producer agent informs the matched consumer agent of the offer. A consumer agent may receive resource offers from more than one producer agents. The consumer agent selects the best offer from all the received resource offers and sends a selection notification to the selected producer agent. The consumer agent sends its jobs to the selected producer agent for execution. If no resource request/offer is received by a producer/consumer agent during the Time To Live (TTL) period of a resource request/offer, then that request/offer is declared as unmatched and is stored in the database by SQL agent. The node agent generates a new resource request/offer after a successful job execution or after the expiry of TTL of its previous resource request/offer.
The experiments are executed in balanced and in task intensive network (TIN) conditions. The task-resource ratio is 50%-50% and 80%-20% in the balanced and TIN respectively. These experiment are executed with 60 nodes for 25 minutes.

5 Experimental Results & Discussion

In this section we discuss the results of our JADE based middleware in terms of transaction price and matchmaking efficiency. The transaction price is calculated

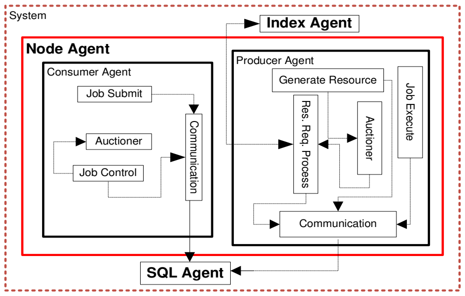
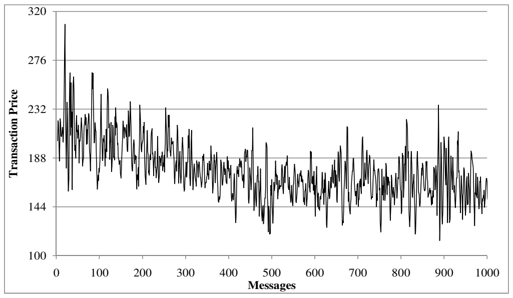
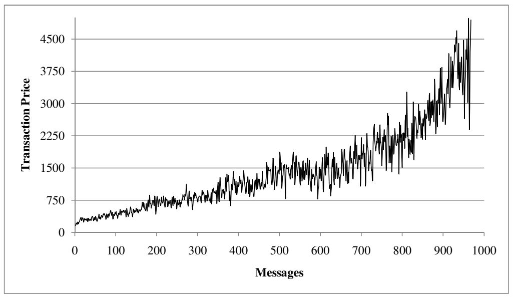


Fig. 1: Overall System Architecture

as the average of ask price and bid price for a matched request and offer pair. The matchmaking efficiency is the ratio of the matched messages to the total message during the simulation. The matchmaking efficiency is 85% and 20% in the balanced and TIN conditions respectively. Since the task-resource ratio is 80-20% in TIN condition, therefore matchmaking efficiency is only 20%. The variation in transaction price is dependent on the network condition. Figure 2a shows the pricing behavior in a balanced network condition. Since ratio of request and offer message is approximately the same, then there is no upward or downward price variation. Whereas in TIN, resources are scarce and tasks are in abundance, still there is a competition among tasks for acquiring resources. The consumer agent(s) keeps on increasing the ask price in order to acquire the requested resources. Therefore, an increasing trend is observed in transaction price in TIN condition (Figure 2b).



(a) Balanced Network



(b) Task Intensive Network

Fig. 2: Experimental Results

6 Conclusions

Actively developed and maintained agent development toolkits that can be used for developing self-managing ad hoc grids and their applications are reviewed in this paper. These toolkits are analyzed for the following factors: autonomy, mobility, interoperability, supported agent communication mechanism, development language, reasoning support, security, FIPA compliance, learning, restiveness, pro-activeness and minimum requirements to develop the application. AgentBuilder, JACK, ZEUS and JADE supported maximum aspects of AC paradigm, therefore, these are most suitable toolkits for AC based applications in general and the autonomic ad hoc grids specially. The JADE toolkit supported all aspects of the AC paradigm due to its extendability with help of add-ons. In future, we would compare other agent development toolkits for resource allocation in a local ad hoc grid.

References

1. Li, Z., Parashar, M.: Rudder: An agent-based infrastructure for autonomic composition of grid applications. Multiagent Grid Systems **1**(3) (2005) 183–195
2. Tesauro, G., Chess, D.M., Walsh, W.E., Das, R., Segal, A., Whalley, I., Kephart, J.O., White, S.R.: A multi-agent systems approach to autonomic computing. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, IEEE Computer Society (2004) 464–471
3. Xiaolin Li, Hui Kang, P.H., Thomas, J.: Autonomic and trusted computing paradigms. In: Proceedings of the 3rd International Conference on Autonomic and Trusted Computing, ATC. (2006) 143–152
4. Horn, P.: Autonomic computing: IBM's perspective on the state of information technology. Technical report, IBM Corporation (October 2001)
5. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE - a white paper. EXP in search of innovation (Special Issue on JADE) (2003)
6. Nwana, H.S., Ndumu, D.T., Lee, L.C., Collis, J.C.: Zeus: a toolkit and approach for building distributed multi-agent systems. In: Proceedings of the third annual conference on Autonomous Agents, AGENTS. (1999)
7. Howden, N., Rönnquist, R., Hodgson, A., Lucas, A.: Jack intelligent agents - summary of an agent infrastructure. In: Proceedings of the 5th International Conference on Autonomous Agents. (2001)
8. Shakshuki, E., Jun, Y.: Multi-agent development toolkits: an evaluation. In: Proceedings of the 17th international conference on Innovations in applied artificial intelligence. (2004) 209–218
9. Horvat, D., Cvetkovic, D., Milutinovic, V.: Mobile agents and java mobile agents toolkits. In: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8, HICSS. (2000)
10. Vrba, P.: Java-based agent platform evaluation. In: Proceedings of the 1st International Conference on Applications of Holonic and Multi-Agent Systems, HoloMAS, Springer-Verlag (2003) 47–58
11. Chmiel, K., Gawinecki, M., Kaczmarek, P., Szymczak, M., Paprzycki, M.: Efficiency of jade agent platform. Scientific Programming **13**(2) (2005) 159–172

12. Collier, R.W.: Agent Factory: A Framework for the Engineering of Agent-Oriented Applications. PhD thesis, University College Dublin, Ireland (2001)
13. FIPA online, http://fipa.org/
14. Tripathi, A.R., Karnik, N.M., Ahmed, T., Singh, R.D., Prakash, A., Kakani, V., Vora, M.K., Pathak, M.: Design of the ajanta system for mobile agent programming. Journal of Systems and Software **62**(2) (May 2002) 123–140
15. ADK home page, http://www.tryllian.org/
16. Acronymics, I.: Agentbuilder: An integrated toolkit for constructing intelligent software agents. Broucher (2004)
17. Marrow, P., Bonsma, E., Wang, F., Hoile, C.: DIET: A scalable, robust and adaptable multi-agent platform for information management. BT Technology Journal **21** (2003) 130–137
18. JADE online, http://jade.cselt.it/
19. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE-java agent development framework. Multi-Agent Programming **15** (2005) 125–147
20. GRAPPA Project, http://ce.et.tudelft.nl/grappa/
21. Helin, H., Laukkanen, M.: Jade goes wireless gearing up agents for the wireless future. EXP in search of innovation (Special Issue on JADE) (2003)
22. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: Implementing a BDI infrastructure for jade agents. EXP in search of innovation (Special Issue on JADE) (2003)
23. Pourebrahimi, B., Bertels, K., Kandru, G., Vassiliadis, S.: Market-based resource allocation in grids. In: 2nd IEEE International Conference on e-Science & Grid Computing. (2006)