A Survey of Agent Platforms

Journal of Artificial Societies and Social Simulation 18 (1) 11
<http://jasss.soc.surrey.ac.uk/18/1/11.html>

Received: 22-Jun-2014  Accepted: 20-Sep-2014  Published: 31-Jan-2015

Abstract

From computer games to human societies, many natural and artificial phenomena can be represented as multi-agent systems. Over time, these systems have been proven a really powerful tool for modelling and understanding phenomena in fields, such as economics and trading, health care, urban planning and social sciences. However, although, intelligent agents have been around for years, their actual implementation is still in its early stages. Since the late nineties many agent platforms have been developed. Some of them have already been abandoned whereas others continue releasing new versions. On the other hand, the agent-oriented research community is still providing more and more new platforms. This vast amount of platform options leads to a high degree of heterogeneity. Hence, a common problem is how people interested in using multi-agent systems should choose which platform to use in order to benefit from agent technology. This decision was usually left to word of mouth, past experiences or platform publicity, lately however people depend on solid survey articles. To date, in most cases multi-agent system surveys describe only the basic characteristics of a few representatives without even providing any classification of the systems themselves. This article presents a comparative up-to-date review of the most promising existing agent platforms that can be used. It is based on universal comparison and evaluation criteria, proposing classifications for helping readers to understand which agent platforms broadly exhibit similar properties and in which situations which choices should be made.

Keywords:
Intelligent Agents, Multi-Agent Systems, Agent Platforms

Introduction

1.1 Multi-agent systems can be used to investigate phenomena or solve problems that are difficult or impossible for humans to study and solve. Actually, they can be used in numerous research areas, from computer games and informatics to economics and social sciences (Schurr et al. 2005; Sun & Naveh 2004; Kaminka 2004; Kubera et al. 2010). Many natural and artificial phenomena can be and in some cases they are represented by multi-agent systems. Hence over time, multi-agent systems have been proven a really powerful tool for modelling and understanding phenomena in fields, such as economics and trading, health care, urban planning and social sciences. Besides, a multi-agent system can be represented as a society, similar to a human society, consisting of entities with characteristics similar to humans, for instance in terms of autonomy and intelligence.

1.2 However, although, intelligent agents have been around for years, their actual implementation is still in its early stages. To this end, the research community has developed a variety of agent platforms in the last two decades; either general purpose or oriented to a specific domain of use. Some of them have already been abandoned whereas others continue releasing new versions. At the same time, the agent-oriented research community is still providing more and more new platforms. All these platforms are as diverse as the community of people who use them. With so many of them available, the choice of which one is best suited for each case was usually left to word of mouth, past experiences or platform publicity. However lately, people interested in using multi-agent systems in order to benefit from agent technology depend increasingly on solid survey articles.

1.3 To date, in most cases multi-agent system surveys describe only some basic characteristics of a few representatives using a variety of criteria, which usually are different from survey to survey (e.g. Gupta & Kansal 2011; Bordini et al. 2006; Nguyen et al. 2002; Nikolai & Madey 2009). Moreover, in most cases surveys so far do not provide case classification, in other words the representatives are just described without indicating relations among them. This article presents a comparative up-to-date review of the existing agent platforms. Our intention was to include and examine the entire continuum of agent platforms that are available today. For this purpose we omitted platforms that can no longer be used and we focused on platforms that are still...
Moreover, this article in order to provide a thorough and comparable description of each agent platform proposes a list of solid universal criteria, which could be used in future work in the field. Additionally, it proposes a number of classifications for helping readers to understand which agent platforms broadly exhibit similar properties and in which situations which choices should be made. For this purpose, we studied as much information as possible for each platform, such as published papers, manual, wikis, websites and forums. Moreover, since each platform is designed for different purposes and domains, we obtained an instance of each platform, installed it and tried to use it following its tutorials and examples, rather than an identical use case that could be proven unfair, in order to study how easy it is to learn and use it. Based on the above research and personal experience, we were finally able to evaluate and compare the platforms. The rest of the article is organized as follows: The next section presents the evaluation criteria in detail. The third section presents a comparative review of agent platforms, also detailing their aspects, while the fourth proposes a number of potential classifications for these agent platforms. Finally, the final section presents the conclusion remarks of the article.

Comparison and Evaluation Criteria

2.1 The high heterogeneity and the vast amount of the available agent platforms is a fact. Hence, choosing the right or most suitable platform for a given problem is still a challenge for the developer. The lack of concrete standardized criteria is an open issue in agent-oriented research that tends to acquire just individual solutions. However, having universal criteria for platform evaluation and comparison has significant advantages. Not only the outcomes of different surveys will be comparable but also comparisons and evaluations during a time interval will reveal the progress that has been or has been not achieved even if the surveys are conducted by different researchers.

In order to present the platforms in a uniform way that facilitates comparison among them and drawing of significant conclusions, we studied past relevant surveys in order to elicit the most used criteria even if they are not explicitly mentioned as such (e.g. Gupta & Kansal 2011; Bordini et al. 2006; Nguyen et al. 2002; Nikolai & Maedy 2009). Additionally, we extensively studied the research disciplines proposed in Braubach et al. (2008) where the authors propose a generic universal criteria catalog for heterogeneous agent development artifacts based on existing standards such as ISO 9126 and ISO 9241 (ISO/IEC 9126-1:2001; ISO9241-110:2006). Based on the above, in this article we present a concrete guideline proposal for comparing and evaluating agent platforms which is directly related to the scope of platform operation and the quality of each platform's provided aspect. This guideline proposal is composed of five discrete criteria categories, namely platform properties, usability, operating ability, pragmatics and security management. Each of these categories has a number of subcriteria that indicate a detailed analysis of each agent platform.

2.3 Platform properties refer to the primary concepts of the platform, describing its basic characteristics that are necessary for a potential user/developer in order to understand the scope and the domain of the platform. Usability refers to the suitability of the platform for the construction of agent applications. Operating ability refers to all these aspects that are taken into account during execution. In other words, operating ability indicates the quality of the platform. Pragmatics refers to external factors that are neither related to the construction nor to the operation of the platform. More specific, pragmatics indicates whether the platform can be used in practice or not. Finally, security management refers to security issues, indicating if the platform is considered safe or not. Table 1 presents the criteria categories along with their subcriteria.

| Platform properties | Usability       | Operating ability | Pragmatics     | Security management |
|---------------------|-----------------|-------------------|----------------|---------------------|
| Developer/Organization | Simplicity    | Performance       | Installation   | End-to-end security |
| Primary domain      | Learnability    | Stability         | User support   | Fairness            |
| Latest release      | Scalability     | Robustness        | Popularity     | Platform security   |
| License             | Standard compatibilities | Programming languages | Technological maturity |                  |
| Open source         | Communication   | Operating systems | Cost           |                     |

2.4 The platform properties and the pragmatics are quite understandable and so there is no need for further explanations. The usability criteria group is also easily understood. Simplicity indicates how simple and understandable the underlying platform's mechanisms are and how easy it is for a developer to use that platform. Learnability refers to whether the platform's mechanisms are familiar to the developer community or not. It is obvious that familiar mechanisms need less or no effort to learn and use while unfamiliar mechanisms could be time and effort consuming. Scalability refers to a potential critical issue; whether the platform can handle varying problem sizes or not. Standard compatibilities, next, are some of the most important features in an agent platform. Standard compliance is an indicator of the platform's flexibility; the more standards a platform adopts the more likely it is available even if their development progress is stopped. Hence, we ended up in twenty four agent platforms.
to cooperate with other applications. Communication is also important since it refers to the type of intra- and/or inter-platform messaging.

2.5 Operating ability criteria group, on the other hand, refers to more specialized issues. Performance is about how efficient is the platform with respect to space and time operations. In this context, we studied how fast the agent communication is and the running speed of the platform itself. Values ranges from average meaning tolerable but not really fast to high meaning excellent platform performance. Stability is about how stable is the platform in case of long term execution whereas robustness is about how tolerant it is in case of breakdowns. In this study, it is a value representing a percentage (based on all available data) of platform crashes, ranging again from average to high (many crashes). Next, programming languages and operating systems indicate the languages and the systems that are used in or by the platform, respectively. Finally, security management is an important aspect in multi-agent systems. The classic case of end-to-end security refers to those protocols and mechanisms that protect access to the platform while fairness refers to those special mechanisms that guarantee equal treatment. Additionally, platform security requires more mechanisms in order to ensure platform security and smooth operation. In this study, platform secure ranges from weak to good, high and strong (in this order) indicating the amount of security mechanisms that are available in the platform.

2.6 However, classic security issues are not enough for the agent community, an aspect that gains more and more attention by the research community is the trust management in the multi-agent systems. Trust models help agents to decide who to trust, encouraging trustworthy behavior and deterring dishonest participation by providing the mean through which reputation and ultimately trust among the community can be quantified (Resnick et al. 2000). In other words, in is not only important to have a secure agent platform but it is required the community acting in that platform to be able to use trust mechanisms. To date, only the EMERALD platform has adopted such mechanisms that support a variety of logics and languages. Hence, we do not include trust management as a separate criterion, although it is important, since it is not supported by the platforms yet except EMERALD.

2.7 Moreover, this article goes one step further. Web is moving towards a true global village, connecting people and knowledge. Eventually, the idea of building a network of content stored on the web making it possible for machines to understand meaning of data and to satisfy requests from people by using it, will came true to its full potential. The semantic wave embraces four stages of internet growth (Mills 2009). The first, Web of Shared Information (Web 1.0), was about connecting information while the current second, Social Web (Web 2.0), is about connecting people. The third, Semantic Web (Web 3.0), has already started and it is about connecting knowledge while the fourth, Web of Intelligence (Web 4.0), will come later and it is about connecting intelligences in a ubiquitous web where both people and things can reason and communicate together.

2.8 The growing complexity and extent of this knowledge however creates the need for more powerful tools. Hence, the question is if agent platforms are or will be able to deal with that. In other words, the question is if agents build in these platforms will be able to traverse the web and integrate the available knowledge. So far, there is no survey, to the best of our knowledge, that takes into account this issue. To this end, we used all the available data to discover which platforms support technologies, semantic web technologies in particular, that can be used for this purpose. As a result, we present an additional classification subsection, where agent platforms are classified according to the degree of their support in semantic web technologies.

Agent Platforms Overview

3.1 This article, as already mentioned, is intended to examine the agent platforms that are available today. Hence, this section presents the platforms that are still available even if their development progress is stopped. An interesting question raises here; why platforms that are no longer developed should be included? The answer is that there are some important and popular agent platforms that are still used by the community though their developers have stopped their maintenance. However, these cases are just a minority. Almost all the presented platforms are active with stable releases. In this context, we have chosen twenty four cases including pure agent platforms, agent frameworks and agent-based simulators because all of them are or will be able to act like agent platforms. Hence, next, each of them is presented in alphabetic order while pivot tables complete the overview at the end of the section.

3.2 First of all, we present the Agent Factory Framework (Russell et al. 2011), an open source collection of tools, platforms, and languages that support the development and deployment of multi-agent systems. The framework is broadly split into two parts: support for deploying agents on laptops, desktops, and servers; and support for deploying agents on constrained devices such as mobile phones and sensors. The former support is realized through Agent Factory Standard Edition (AFSE), and the latter support is realized through Agent Factory Micro Edition (AFME), a light weight agent platform that has been designed to work with J2ME-CDLC MIDP2.0. Currently, Agent Factory is used in a number of projects including mobile computing, robotics and other, e.g. the MiRA- Mixed Reality Agents project (Holz et al. 2011).

3.3 AgentBuilder (Acronymics Inc. 2004) is another case of traditional agent platform that can be used in numerous simulation cases. It is a KQML-compliant integrated tool suite for constructing intelligent software agents, allowing software developers with no background in intelligent systems or intelligent agent technologies to quickly and easily build intelligent agent-based applications. KQML is a language and protocol for communication among software agents and knowledge-based systems (Finin et al. 1994). AgentBuilder is quite simple and easy to use even for unfamiliar users is its main advantage. AgentBuilder is an
3.4 **AgentScape** (Oey et al. 2010) is an agent platform, on the other hand, has been designed to support the design and deployment of large-scale, heterogeneous, secure, distributed agent systems. Within AgentScape, agents are active entities that reside within locations, communicate with each other and access services. The AgentScape approach to management is targeted to scalability and autonomicity. The above allows AgentScape to be a promising agent platform for studies including large-scale distribution and heterogeneity.

3.5 **AGLOBE** (Sislak et al. 2006) is an agent platform designed for testing experimental scenarios featuring agents' position and communication inaccessibility, but it can also be used without these extended functions. The platform provides functions for the residing agents, such as communication infrastructure, store, directory services, migration function, deploy service, etc. Communication in AGLOBE is very fast and the platform is relatively lightweight. AGLOBE platform is not yet fully compliant with the Foundation for Intelligent Physical Agents’ (FIPA) specifications, e.g. it does not support inter-platform communication (communication with other agent platforms is not supported yet). This interoperability is not necessary when developing closed systems, where no communication outside these systems is required (e.g. agent-based simulations). AGLOBE is sponsored by US Air Force and it is suitable for real-world simulations including both static (e.g. towns, ports, etc.) and mobile units (e.g. vehicles). In such case the platform can be started in extended version with Geographical Information System (GIS) services and Environment Simulator (ES) agent.

3.6 **AnyLogic** (Borshchev 2013) is a multi-method simulation platform that supports not only agent-based general purpose simulations but also it supports system dynamics and process-centric (discrete event) modeling. The flexibility of its modeling language enables the user to capture the complexity and heterogeneity of business, economic and social systems to any desired level of detail. AnyLogic's graphical interface, tools, and library objects allow quick modeling of diverse areas such as manufacturing and logistics, business processes, human resources, consumer and patient behavior. Additionally, the object-oriented model design paradigm supported by AnyLogic provides for modular, hierarchical, and incremental construction of large models. It is probably one the best solutions for simulation if there is no extensive coding background to start with.

3.7 **Cormas** (Le Page & Bousquet 2007) is a simulation platform based on the VisualWorks programming environment which allows the development of applications in the Smalltalk object oriented language. Cormas pre-defined entities are Smalltalk generic classes from which, by specialization and refining, users can create specific entities for their own model. It facilitates the construction of agent-based models and the design, monitoring and analyzing of agent-based simulation scenarios. Cormas was primarily oriented towards the representation of interactions between stakeholders about the use of natural renewable resources. Cormas is very popular and it provides some interesting features for real life applications.

3.8 **Cougar** (Helsinger & Wright 2005) follows a Cognitive Agent Architecture and is a DARPA-funded open-source agent platform that offers special support for logistics problems. The platform is not FIPA-compliant. It facilitates the development of agent based applications that are complex, large scale and distributed. Cougar's cognitive architecture is another promising feature since the majority of the available platforms does not handle with such issues despite the fact that they are important for simulating human thinking and acting.

3.9 **CybelePro** (CybelePro 2006) provides its users with a robust high-performance infrastructure for rapid development and deployment of large-scale, high performance agent-based systems. CybelePro is the commercial release of Intelligent Automation Inc. Cybele agent infrastructure has been used extensively by the government, industry and academia for applications such as military logistics, modeling, simulation and control of air and ground transportation, communication networks and a development of open systems.

3.10 **EMERALD** (Kravari et al. 2010) is a quite new implementation framework for interoperable reasoning among agents in the Semantic Web, by using third-party trusted reasoning services. The advantage is that every agent can exchange its position justification arguments with any other agent, without the need for all agents to conform to the same kind of rule paradigm or logic. It is built on top of JADE (presented below) and it is fully FIPA-compliant. Moreover, EMERALD was involved in cross-community interoperations such as in Kravari et al. (2012). It supports a variety of logics and languages such as Java, JESS, XML, RDF, RuleML and Prolog. Additionally, EMERALD is the only agent platform that supports trust and reputation mechanisms in order to support trustworthiness and efficient decision making in the multi-agent system. It has been used so far in studying how agents act on behalf of their users in cases such as trading.

3.11 **GAMA** (Grignard et al. 2013) is a simulation platform, which aims at providing field experts, modelers, and computer scientists with a complete modeling and simulation development environment for building spatially explicit multi-agent simulations. It provides capabilities concerning the tight combination of 3D visualization, GIS data management, and multi-level modeling. GAMA supports capabilities for building large models written in the GAML agent-oriented language, with an optional graphical modeling tool to support rapid design and prototyping. Users are also able to instantiate agents from GIS data, databases or files and executing large-scale simulations (up to millions of agents) and design rich user interfaces that support deep inspections on agents, user-controlled actions and panels, and multiple multi-layer 2D/3D displays and aspects.

3.12 **INGENIAS Development Kit** (Gómez-Sanz & Pavón 2004) is a tool for developing Multi-Agent Systems that supports the INGENIAS methodology. INGENIAS promotes a model driven approach based on the use of INGENME (INGENIAS Meta-Editor), a tool for producing self-contained visual editors for languages defined using an XML file. INGENME is used to produce a
visual editor for Multi-Agent Systems where the systems' specifications are processed to produce programming code, html documents, or other required products. INGENIAS addresses roundtrip engineering issues as well, by a concrete folder structure and a code-to-specification information migration tool.

3.13 **JACK** (Winikoff 2005) is a mature, cross-platform environment for building, running and integrating commercial-grade multi-agent systems. It is built on a sound logical foundation: BDI (Beliefs/Desires/Intentions). BDI is an intuitive and powerful abstraction that allows developers to manage the complexity of the problem. It is entirely written in Java. JACK is highly portable and runs on anything from smartphones to high-end, multi-CPU servers. Its Java foundation means that JACK can run with multiple threads across multiple CPUs, has platform-independent GUIs, and is easily integrated with third-party libraries. JACK is the leading edge commercial BDI-agent toolkit that uses an intuitive language that extends the Java programming language with certain agent specific keywords. It is the first platform with support for capabilities and generic team structures, but does not yet support the FIPA specifications.

3.14 **JADE** (Bellifemine et al. 2003) is a framework fully implemented in Java. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications. The agent platform can be distributed across machines (which do not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in the Java language and the minimal system requirement is version 1.2 of JAVA (the run time environment or the JDK), hence it can be adapted to be used on devices with limited resources such as mobile phones. JADE is industry-driven and currently it is the most popular FIPA-compliant agent platform in academic and industrial community. It is a free, open source and stable software distributed by Telecom Italia, the copyright holder. Since May 2003, a JADE Board has been created that supervises the management of the JADE Project. Currently the JADE Board lists 5 members: Telecom Italia, Motorola, Whitestein Technologies AG, Profactor GmbH, and France Telecom R&D.

3.15 **Jadex** BDI Agent System (Braubach & Pokahr 2013) follows the Belief Desire Intention (BDI) model and facilitates easy intelligent agent construction with sound software engineering foundations. It allows for programming intelligent software agents in XML and Java. The Jadex research project is conducted by the Distributed Systems and Information Systems Group at the University of Hamburg. The developed software framework is available under GNUs LGPL license, and is continuously evolving. Jadex has been put into practice in the context of several research, teaching, and industrial application scenarios some of which are described in its website. It has been used to build applications in different domains such as simulation, scheduling, and mobile computing. For example, Jadex was used to develop a multi-agent application for negotiation of treatment schedules in hospitals (Braubach et al. 2014). In the latest version, the programming model of Jadex is based on the notion of active components that are conceptually based on SCA (service component architecture). This allows for designing an application as hierarchical decomposition of components interacting via services and thus helps making complexity controllable. Active components extend SCA in several directions as it is intended to work in concurrent and dynamic distributed systems.

3.16 **JAMES II** (Java Framework for Modeling & Simulation) (Himmelspach & Uhrmacher 2007) is a pure Java framework, with no external dependencies, for modeling and simulation. It is based on a plug-in architecture, shortly named "plug'n simulate", built for maximum flexibility. The architecture allows any modeling and simulation technique to be integrated into the framework via plug-ins. Moreover, it provides a solid foundation of abstractions, algorithms, workflows and tools, focusing on efficiency. In this context, JAMES II provides hundreds of plug-ins, allowing automatic selection from the available list of alternative plug-ins. Yet, it does not enforce any reuse of its parts but allows its users to choose which functionality they want to apply. Additionally, JAMES II allows full control over experiment types and parameters, such as parameter scans, optimization, computation end policies and number of replications. So far, JAMES II has already been used in various application areas, from small notebook-based to large cluster-based experiments.

3.17 Although **JAS**, the Java Agent-Based Simulation Library (Sonnessa 2004) is not a pure agent platform, it acts as such. JAS is a simulation toolkit specifically designed for agent-based simulation modeling. JAS is a Java clone of the Swarm library originally developed by researchers at the Santa Fe Institute. The core of the JAS toolkit is its simulation engine based on the standard discrete event simulation paradigm, which allows time to be managed with high precision and from a multi-scale perspective. Many features of JAS are based on open source third party libraries. The core of the JAS toolkit is represented by the simulation engine. It is based on the standard discrete-event simulation paradigm, which allows managing the time with high precision and multi-scale perspective. Thanks to its discrete-event engine, JAS represents a good compromise in simulating both discrete and continuous agent-based models. This makes JAS a generic discrete-event simulation toolkit, also useful to realize process workflow simulation models.

3.18 **Jason** (Bordini et al. 2007) is a fully-fledged interpreter for an extended version of AgentSpeak, a BDI agent-oriented logic programming language, and is implemented in Java. It implements the operational semantics of that language, and provides a platform for the development of multi-agent systems, with many user-customizable features. Using SACI, a multi-agent system can be distributed over a network effortlessly. Some of the features available in Jason are: speechact based inter-agent communication, annotations on plan labels, fully customizable (in Java) selection functions, trust functions, and overall agent architecture, straightforward extensibility by means of user-defined internal actions, a clear notion of a multi-agent environment, which is implemented in Java (this can be a simulation of a real environment).

3.19 **J4AC**, the Java-based Intelligent Agent Componentware (Hirsch et al. 2009) is a Java-based agent architecture and framework
that eases the development and the operation of large-scale, distributed applications and services. The framework supports the design, implementation, and deployment of software agent systems. The entire software development process, from conception to deployment of full software systems, is supported by JIAC. It also allows for the possibility of reusing applications and services, and even modifying them during runtime. The focal points of JIAC are distribution, scalability, adaptability and autonomy.

3.20 **MaDKit**, the Multiagent Development Kit (Gutknecht & Ferber 2000) is an open source modular and scalable multiagent platform written in Java and built upon the AGR (Agent/Group/Role) organizational model. MaDKit agents play roles in groups and thus create artificial societies. It provides general agent facilities, such as lifecycle management, message passing and distribution, and allows high heterogeneity in agent architectures and communication languages, and various customizations.

3.21 **NetLogo** (Tisue & Wilensky 2004) is a multi-agent programmable modeling environment. It is designed, in the spirit of the Logo programming language, to be "low threshold and no ceiling". NetLogo enables exploration of emergent phenomena; it comes with an extensive models library including models in a variety of domains, such as economics, biology, physics, chemistry, psychology and system dynamics. It can be used both by teachers in the education community and domain experts without a programming background to model related phenomena. Beyond exploration, NetLogo allows authoring of new models and modification of existing models. It also powers HubNet, a technology that uses NetLogo to run participatory simulations in the classroom. For instance, in a participatory simulation, a whole group of users takes part in enacting the behavior of a system, e.g. acting as a separate, independent agent. NetLogo is very popular in the education and research community.

3.22 **MASON** (Luke et al. 2005) is a fast, easily extendable, discrete event multi-agent simulation toolkit in Java acting as a simulation agent platform. It was designed to serve as the basis for a wide range of multi-agent simulation tasks ranging from swarm robotics to machine learning to social complexity environments. MASON carefully delineates between model and visualisation, allowing models to be dynamically detached from or attached to visualisers and enabling check pointing.

3.23 The **Repast Suite** (North et al. 2007), on the other hand, is a family of advanced, free, and open source agent-based modeling and simulation platforms that have collectively been under continuous development for over 14 years. Repast was initially developed at the University of Chicago's Social Science Research Computing Lab. It is a widely used free and open-source, cross-platform, agent-based modeling and simulation toolkit, providing multiple implementations in several languages and many built-in adaptive features. Currently, there are two editions of Repast and several ways to models in each edition aiming at satisfying many different kinds of users and cases.

3.24 **SeSAm** (Shell for Simulated Agent Systems) (Klugl 2009) is a generic environment for the development and simulation of Multi-Agent models. Its main focus is to enable scientists to construct models by visual programming, since the agent paradigm is very intuitive, especially when modeling societies. The main entities in a SeSAm model are agents, resources and the world. There are some aspects that allow scaling up for complex multi-agent simulation: user functions, user features and model-specific data types. Simulation runs may be executed for different situations and aggregated into so called experiments. Also model instrumentation for gathering and visualizing simulation data is possible via the so called analysis. Before starting a simulation run, the model is compiled using standard optimization techniques from compiler theory, thus the power of visual programming was combined with fast execution. SeSAm was first a Lisp-based system but since 2000 it is redesigned and implemented in Java.

3.25 Finally, **Swarm** (Swarmfest 2012) the first re-usable software tool created for agent based modeling and simulation was developed at the Santa Fe Institute in 1994. Swarm was specifically designed for artificial life applications and studies of complexity. It was originally written in Objective-C, and then ported to Java. Swarm was originally developed for multi agent simulation of complex adaptive systems.

3.26 Next, five tables present the values of the described criteria (section 2) for each of the above agent platforms. The basic characteristics of each platform are presented in Table 2. The usability characteristics of each agent platform are presented in Table 3, whereas Table 4 presents the operating characteristics. Pragmatics of each agent platform is presented in Table 5 while, finally, information about the security management in each agent platform is presented in Table 6. The two last columns of that table include also the main publication of each agent platform and a download link for quick reference and space saving purposes.

| Table 2: Platform properties |
|-----------------------------|
| **Name** | **Developer/Organization** | **Primary domain** | **Latest release** | **License** | **Open source** |
| --- | --- | --- | --- | --- | --- |
| 1. Agent Factory | University College Dublin | General purpose | AFSE v3.0 (07/12/2011) | LGPL | Yes |
| 2. AgentBuilder Lite | Acronyms Inc. | General purpose | AgentBuilder Lite/Pro v1.4 (06/11/2004) | Proprietary, Discounted academic licenses | No |

http://jasss.soc.surrey.ac.uk/18/1/11.html 6 20/10/2015
|   | Project | Organization | Description | Version | License | Open Source? |
|---|---------|--------------|-------------|---------|---------|-------------|
| 3. | AgentScape | Delft University of Technology | Large-scale distributed agent systems | AgentScape2 | BSD | Yes |
| 4. | AGLOBE | Czech Technical University | Real-world simulations | AGLOBE v5.5 | LGPL | Yes |
| 5. | AnyLogic | The AnyLogic Company | General purpose, distributed agent based simulations | AnyLogic 7.0.1 | Commercial, Academic | No |
| 6. | Cormas | Cirad research centre | Natural resources and agent-based simulations | Cormas2014 | GPL | Yes |
| 7. | Cougaar | Raytheon BBN Technologies | Complex, large-scale, distributed applications | Cougaar 12.7 | Open Source License | Yes |
| 8. | CybelePro | Intelligent Automation Inc | Large-scale distributed systems | CybelePro 3.1.3 | Commercial, Academic | No |
| 9. | EMERALD | LPIS Group, Aristotle University of Thessaloniki | Distributed applications composed of autonomous entities | EMERALD v1.0 | GPL | Yes |
| 10. | GAMA | IRD/UPMC International Research Unit UMMISCO | Large-scale distributed spatially explicit agent-based simulations | GAMA 1.6.1 | GPL | Yes |
| 11. | INGENIAS Development Kit | Universidad Complutense Madrid | General purpose agent based | IDK 2.8 | CC By-SA | Yes |
| 12. | JACK | AOS | Dynamic and complex environments | JACK 5.5 | Commercial, Academic | No |
| 13. | JADE | Telecom Italia (TILAB) | Distributed applications composed of autonomous entities | Jade 4.3.1 | LGPLv2 | Yes |
| 14. | Jadex | Hamburg University | Distributed applications composed of autonomous BDI entities | Jadex 2.4.0 | LGPLv2 | Yes |
| 15. | JAMES II | University of Rostock | General purpose agent based modeling and simulations | JAMES II 0.9.6 | JAMESLIC (compatible with GPL) | Yes |
| 16. | JAS | Università di Torino | General purpose agent based | JAS 1.2.1 | LGPL; 3\textsuperscript{rd} party licenses | Yes |
| 17. | Jason | Universidades of Rio Grande do Sul and Santa Catarina | Distributed applications composed of autonomous BDI entities | Jason 1.4.0 | LGPLv2 | Yes |
| 18. | JIAC | Technische Universität Berlin | Large-scale distributed systems | JIACv 5.1.5 | Apache License V2 | Yes |
| 19. | MaDKit | Institut universitaire de technologie | Multi-agent systems with agent based simulation | MaDKit 5.0.5 | GPL | Yes |
| 20. | MASON | George Mason | Event-driven | MASON v17 | Academic | Yes |
| No. | Name                  | University                                                                 | Type                      | Version                | License   | Supported?: |
|-----|-----------------------|----------------------------------------------------------------------------|---------------------------|------------------------|-----------|-------------|
| 21. | NetLogo               | The Center for Connected Learning (CCL) and Computer-Based Modeling, Northwestern University | Agent-based simulations   | NetLogo 5.0.5 (19/12/2013) | GPL       | No          |
| 22. | Repast                | University of Chicago                                                     | Agent-based simulations   | Repast Simphony2.1 (12/08/2013) | New BSD   | Yes         |
| 23. | SeSAm                 | Örebro University                                                         | Agent-based simulations   | SeSAm 2.5.1 (02/05/2012) | LGPL      | Yes         |
| 24. | Swarm                 | Swarm Development Group                                                   | General purpose agent based | Swarm 2.2 (12/02/2005) | GPL       | Yes         |

Table 3: Usability overview

| Name                  | Simplicity                                                                 | Learnability | Scalability | Standard compatibility | Communication |
|-----------------------|----------------------------------------------------------------------------|--------------|-------------|-------------------------|---------------|
| 1. Agent Factory      | Simple/Poor in features GUI                                               | Average      | Good        | Partially FIPA (agent lifecycle) | HTTP          |
| 2. AgentBuilder       | Simple/Poor in features GUI                                               | Easy         | Good        | KQML                    | KQML, CORBA and TCP/IP |
| 3. AgentScape         | Simple/Poor in features GUI                                               | Average      | Good        | None known              | Message exchange (platform defined syntax) |
| 4. AGLOBE              | Simple/Poor in features GUI                                               | Easy         | High        | Partially FIPA (ACL), GIS | ACL no interplatform |
| 5. AnyLogic           | Average (nor simple neither complicated) interface/Rich in features GUI    | Easy         | High        | GIS, 3D capabilities    | Message exchange (platform defined syntax) |
| 6. Cormas              | Simple/Some useful for simulation GUI features                            | Easy         | Good        | None known              | peer-to-peer  |
| 7. Cougaar             | Complicated interface/Some unfamiliar GUI features                        | Average      | High        | None known              | Cougaar Message Transport Service (MTS) |
| 8. CybelePro           | Average (nor simple neither complicated) interface/ Some useful for simulation GUI features | Average      | High        | None known              | Message exchange (platform defined syntax) |
| 9. EMERALD             | User-friendly, useful GUI, many familiar features (based on JADE platform) | Easy         | High (on JADE) | FIPA, Semantic Web standards (see Table 4) | ACL (Asynchronous) |
| 10. GAMA               | Simple/Some useful for simulation GUI features                            | Easy         | Good        | FIPA, (GIS, 3D capabilities) | ACL (GAML messages that represent FIPA) |
|   | Operating ability |
|---|------------------|
| 11. | INGENIAS Development Kit | Simple/User friendly GUI editor | Easy | Good | AUML | ACL messages) peer-to-peer |
| 12. | JACK | User friendly/ Rich in features GUI | Easy | High | FIPA | DCI network, TCP/IP |
| 13. | JADE | User-friendly, useful GUI, many familiar features | Easy | High | FIPA, CORBA | ACL (Asynchronous), MTPs, RMI, IIOP, HTTP, WAP |
| 14. | Jadex | User-friendly GUI/Many features | Average | High | FIPA, SOA, WSDL | HTTP |
| 15. | JAMES II | Simple/ User friendly GUI editor | Easy | High | None known | Depend on plugins (e.g. via time-stamped messages) |
| 16. | JAS | Simple/Poor in features GUI | Easy | Average | Partially FIPA (agent lifecycle), GraphML, XML | Message exchange (platform defined syntax) |
| 17. | Jason | Simple/Some familiar user-friendly GUI features | Easy | Good | Partially FIPA (language) | speech-act based, KQML |
| 18. | JIAC | Average (nor simple neither complicated) interface/ Some useful for distributed systems GUI features | Average | High | None known | ActiveMQ |
| 19. | MaDKit | Complicated interface (many pop ups)/Rich in features GUI | Average | Good | UML | peer-to-peer |
| 20. | MASON | Complicated interface/ Useful features for multiple applets | Average | Average | None known | Message exchange (event-driven, platform defined syntax) |
| 21. | NetLogo | Simple/ Some useful for simulations features | Easy | Good | None known (yet there are extensions for GIS and 3D capabilities) | Extensions for message exchange (extensions' defined syntax) |
| 22. | Repast | Simple/ Limited GUI features | Easy | Good | None known | peer-to-peer |
| 23. | SeSAm | Simple/Rich in features GUI | Easy | Good | None known | Message exchange (platform defined syntax) |
| 24. | Swarm | Complicated interface/ Limited GUI features | Average | Average | None known | Message exchange (platform defined syntax) |

Table 4: Operating ability of each agent platform
| Name            | Performance | Stability | Robustness | Programming languages | Operating systems     |
|-----------------|-------------|-----------|------------|-----------------------|-----------------------|
| Agent Factory   | Good        | Good      | Average    | Java, AFAPL, AgentSpeak | Any with JVM           |
| AgentBuilder    | Good        | High      | High       | KQML, Java, C,C++     | Windows, Linux, Sun Solaris |
| AgentScape      | Good        | Good      | Good       | Java (plus use of XML) | Any with JVM           |
| AGLOBE          | High        | High      | High       | Java                  | Any with JVM           |
| AnyLogic        | High        | High      | High       | Java, UML-RT (UML for real time) | Any with JVM           |
| Cernos          | High        | Good      | Good       | SmallTalk             | Win, Linux             |
| Cougaar         | High        | High      | High       | Java                  | Win, Linux             |
| CybelePro       | High        | High      | High       | Java                  | Any with JVM           |
| EMERALD         | High (on JADE) | High      | High       | Java, JESS, RuleML, Prolog (plus use of XML, RDF) | Any with JVM           |
| GAMA            | Good        | Good      | Good       | GAML                  | Mac OS X, Windows, Linux |
| INGENIAS        | Good        | Good      | Good       | Java (plus use of XML) | A Any with JVM         |
| JADE            | High        | High      | High       | Java, JACK Agent Language (JAL) (plus use of XML) | Windows, Macintosh, Unix, generic Java, iPAQ |
| Jadex           | High        | Good      | High       | Java (plus use of XML) | Any with JVM           |
| JAMES II        | High        | High      | High       | Java                  | Any with JVM           |
| JAS predecessor  | Good        | Average   | Low        | Java                  | Any with JVM           |
| Jason           | Good        | High      | Good       | Java, AgentSpeak      | Windows, MacOS, Linux  |
| JIAC            | High        | High      | High       | Java (plus use of XML) | Any with JVM           |
| MaDKit          | Good        | Good      | Good       | Java, C/C++, Python   | Any with JVM           |
| MASON           | Good        | Good      | Average    | Java                  | Win                   |
| NetLogo         | Good        | Good      | Average    | NetLogo               | Any with JVM           |
| Repast          | High        | High      | High       | Java, C#, C++, Lisp, Prolog, Python | Any with JVM           |
| SeSAm           | Good        | High      | Good       | Java (plus plugin for ontologies) | Any with JVM           |
| Swarm           | Average     | Average   | Low        | Java                  | Win, Linux             |

Table 5: Pragmatics overview
| #  | Software       | Features                                                                 | Stability | Release    | Licensing                                                                 |
|----|----------------|---------------------------------------------------------------------------|-----------|------------|---------------------------------------------------------------------------|
| 2. | AgentBuilder   | Plenty available installers Good (Consulting training, example, FAQ, docs, defect reporting, mailing list) | Medium    | Old but Stable release | AgentBuilder Lite $99 AgentBuilder Pro $925-$4,525 |
| 3. | AgentScape     | Installer Good (forum, email, docs)                                       | Low       | Stable     | Free                                                                      |
| 4. | AGLOBE         | Command line usage Average (docs, mail contact)                           | Medium    | Stable     | Free                                                                      |
| 5. | AnyLogic       | Installer High (extensive documentation and supporting tools)             | Medium    | Stable     | AnyLogic Advanced $6,199 Professional $15,800 University Researcher License $3,500 Educational Licenses $485 |
| 6. | Cormas         | Command line usage Good (forum, email, docs)                             | Medium    | Latest release beta | Free                                                                      |
| 7. | Cougaar        | Eclipse plug-in Good (FAQ, docs, forums, mailing lists)                   | Low       | Stable     | Free                                                                      |
| 8. | CybelePro      | Installers Good (sales support)                                          | Low       | Stable     | Commercial $1000-$4000, Academic $600-$2400                               |
| 9. | EMERALD        | Installer Average (documentation, mail contact)                          | Low       | Stable     | Free                                                                      |
| 10. | GAMA           | Installer Good (forum, email, docs, wiki, social media)                  | Low       | Stable     | Free                                                                      |
| 11. | INGENIAS       | Development Kit Command line usage Average (docs, mail contact)          | Medium    | Stable     | Free                                                                      |
| 12. | JACK           | Plenty available installers High (extensive documentation and supporting tools) | High     | Stable     | Unknown, Free trial version                                               |
| 13. | JADE           | Command line usage High (FAQ, mailing list, defect list, API, docs)      | High (most popular) | Stable | Free                                                                      |
| 14. | Jadex          | With the appropriate jar file Average (docs, mail contact)               | High      | Stable     | Free                                                                      |
Table 6: Security management overview

| Security management | Name         | End-to-end security | Fairness | Platform security | Publication | Available at |
|---------------------|--------------|----------------------|----------|-------------------|-------------|--------------|
| 1.                  | Agent Factory | Signature and Encryption | No       | Average            | Russell et al. 2011 | Agent Factory 2014 |
| 2.                  | AgentBuilder  | No                   | No       | Weak              | Acronymics Inc. 2004 | AgentBuilder 2014 |
| 3.                  | AgentScape    | Authentication, private logging | Yes     | Good              | Oey et al. 2010 | AgentScape 2014 |
| 4.                  | AGLOBE        | None known           | No       | Strong (closed system) | Sislak et al. 2006 | AGLOBE 2014 |
| 5.                  | AnyLogic      | Authentication       | Yes      | Strong (closed system) | Borshchev 2013 | AnyLogic 2014 |
| 6.                  | Cormas        | No                   | No       | Strong (closed system) | Le Page & Bousquet 2007 | Cormas 2014 |
| 7.                  | Cougaar       | Authentication       | No       | High              | Helsinger & Cougaar |            |
Agent Platforms Classification

4.1 Developers and users need help in order to distinguish which agent platforms broadly exhibit similar properties and to understand in which situations which choices should be made. To this end, this section presents some classifications that answer the most common questions; e.g. which platforms uses the language I need or know, which is the most suitable for my case, etc., based on various classification criteria.

Programming language

4.2 Programming languages are important because different languages have different implications in terms of expressiveness, ease of programming, portability, and compatibility. Usually, agent platforms are written in C, C++ or Java but there are various programming languages that may be used to program a multi-agent system. Many of these languages were stemmed from a need for specialization and they were created and used in the context of a specific agent platform. Here (Table 7) we refer only to languages that are used to program a multi-agent system built on top of the platform and not those that were used to program the platform itself. It is worth mentioning that the main programming language adopted by multi-agent systems is Java. Almost 80% of the platforms employ Java as their primary programming language while about 20% of the platforms use C++ to program...
models and only 10% use C. Mention that we used XML in the classification although it is not a pure programming language since it is used in many simulation cases for knowledge representation and communication purposes.

4.3 Hence, whenever the language is criterion for platform selection, the programming language classification presented here could be a useful tool. For instance, if there is a need for a general-purpose, concurrent and object-oriented language with just a few implementation dependencies then platforms using Java should be chosen. On the other hand, if the developing system/simulation should run without external applications (like the Java virtual machine – JVM) then platforms using C/C++ should be chosen. Moreover, C++ was designed for infrastructure programming hence platforms using it are suitable for that kind of studies. In cases where researchers want to describe what the system should accomplish in terms of the problem domain, rather than how to accomplish it as a sequence of the programming language primitives the platforms that support declarative/rule programming languages should be used.

Table 7: Programming language overview

| Language Type          | Platforms                                                                 |
|------------------------|---------------------------------------------------------------------------|
| Java                   | JADE, SeSAm, Jadex, JAS, AgentBuilder, EMERALD, Repast, MaDKit, CybelePro, Cormas, AGLOBE, Cougaar, Swarm, MASON, INGENIAS Development Kit, AnyLogic, JAMES II |
| C/C++                  | AgentBuilder, Swarm (Objective C), Repast (plus C#), MaDKit               |
| Declarative/rule       | Repast (Lisp, Prolog), JADE (JESS), EMERALD (JESS)                        |
| Programming            |                                                                           |
| Python                 | Repast, MaDKit                                                           |
| AgentSpeak             | Jason, Agent Factory (AFSE)                                               |
| SmallTalk              | Cormas                                                                   |
| JAL                    | JACK                                                                     |
| NetLogo                | NetLogo                                                                  |
| GAML                   | GAMA                                                                     |
| XML                    | AgentScape, EMERALD, INGENIAS Development Kit, JACK, Jadex, JIAC          |
| Multiple languages     | AgentScape, EMERALD, INGENIAS Development Kit, JACK, Jadex, JIAC, Jason, MaDKit, Repast, AgentBuilder, AgentFactory |

FIPA compliance

4.4 The Foundation for Intelligent Physical Agents (FIPA) is an international organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications. FIPA-compliance is a reference point for many researchers. Some platforms are fully compliant whereas others adopt only some of the available specifications. For instance, JAS and Agent Factory platforms supports only those specification that refer to the agent lifecycle whereas AGLOBE and Jason support specifications that refer to the communication principles and language, ACL and KQML respectively. Table 8 presents a FIPA-compliance overview for the above studied agent platforms.

Table 8: FIPA compliance overview

| Compliance Level | Platforms                                                                 |
|------------------|---------------------------------------------------------------------------|
| Full compliance  | JADE, Jadex, JACK, EMERALD                                                |
| Partial compliance | JAS, Jason, AGLOBE, Agent Factory, SeSAm, GAMA                          |
| No compliance    | Cougaar, Swarm, MASON, INGENIAS Development Kit, Cormas, Repast, MaDKit, CybelePro, JIAC, AgentScape, AnyLogic, NetLogo, JAMES II |

Application Domain

4.6 Some agent platforms are general purpose and can be used for a variety of applications domains while others are domain specific. Having a specialized platform means that it was designed for that domain and it is more efficient to be used there.
However, some of these platforms can be used for many other domains. The following table (Table 9) presents the various domains and the agent platforms that are specialized or commonly used for them.

4.7 To this end, the following classification indicates which platform could be more suitable for a preferred domain. For instance, if the case is a social study then SeSAm or Repast would be better choices. On the other hand, GIS systems are better studies using platforms such as AGLOBE, Cougar or SeSAm. Moreover, there are platforms that can be used in almost every domain like JADE, which is actually the most popular agent platform. However, usually there are more than one choices (platforms) for each domain, hence the final choice depends on a combination of the desirable platforms characterizes. For instance, if there is a need for a general purpose platform that uses C/C++ then MaDKit should be used (combination of Tables 7 and 9 data).

Table 9: Platforms’ application domains overview

| General purpose distributed simulations | JADE, Jadex, Jason, EMERALD, MaDKit, CybelePro, JIAC, AgentScape, AnyLogic, GAMA |
| General purpose agent based simulations | JAS, AgentBuilder, Agent Factory (AFSE), Swarm, MASON, INGENIAS Development Kit, Repast, MaDKit, AgentScape, SeSAm, AnyLogic, NetLogo, GAMA, JAMES II |
| Scientific simulations | Swarm, MASON, Cormas, Repast (Social Sciences), MaDKit, CybelePro, SeSAm, AnyLogic, GAMA |
| Dynamic and complex environments | JACK, Cougar, CybelePro, AnyLogic |
| Real - world - GIS | AGLOBE, Cougar (integrated with OpenMap), Repast, CybelePro, SeSAm, AnyLogic, GAMA |
| Large scale simulations | Cougar, CybelePro, JIAC, AgentScape, GAMA, JAMES II |
| Scheduling & planning | Jadex, CybelePro, SeSAm, AnyLogic, NetLogo, GAMA |
| Mobile computing | JADE, Jadex, Agent Factory |
| Multiple domains | JADE, Jadex, MaDKit, CybelePro, Cougar, JIAC, AgentScape, Agent Factory, Repast, SeSAm, AnyLogic, GAMA, JAMES II |
| Artificial life and behavioral observation | JADE, Jadex, Jason, SeSAm, EMERALD, JACK, Cougar, CybelePro, AnyLogic, NetLogo, GAMA |
| Biological & social studies | JADE, SeSAm, Jadex, Jason, EMERALD, JACK, Cougar, CybelePro, MaDKit, Repast, AnyLogic, NetLogom, GAMA, JAMES II |
| Economics/eCommerce | JADE, EMERALD, JACK, Cougar, CybelePro, MaDKit, AnyLogic |
| Natural resources & environment | Cormas, Swarm, SeSAm, AnyLogic, NetLogo, GAMA, JAMES II |

Semantic Web Technologies

4.8 Furthermore, as already mentioned, our intention is to study which agent platforms can be used in the Web, hence we tried to figure out which of them support semantic web technologies. The potential of a platform however has nothing to do with its current maturity in semantic web technologies. EMERALD for instance is fully semantic web oriented while Jadex support some of these technologies, e.g. the XML language. Hence, EMERALD can instantly be used in web applications whereas Jadex has first to be more extended. At the same time, there are plenty of the platforms that do not support yet semantic web technologies but their architecture could be easily extended to that direction. Cormas, for instance, is not yet semantic web oriented but an extension of it where semantic web communication languages will be supported would reveal how agents would act under real situations on behalf of their users. Table 10 presents a classification of the above platforms according to the degree (High, Average, Low) of their support in semantic web technologies (see Tables 3 and 4). Platforms with an average degree of support usually have adopted one or more semantic web languages (usually XML), for instance SeSAm supports ontologies (the only one), while those with a low degree have poor or no support at all.

Table 10: Semantic Web technologies support

| Degree   | Platforms |
|----------|-----------|
| High     | EMERALD   |
| Average  | Jadex, JACK, JIAC, AgentScape, INGENIAS Development Kit, SeSAm |
| Low      | JADE, CybelePro, Cougar, AgentBuilder, AGLOBE, Cormas, Jason, JAS, Agent Factory, Swarm, Repast, MASON, MaDKit, INGENIAS Development Kit, AnyLogic, NetLogo, GAMA, JAMES II |

http://jasss.soc.surrey.ac.uk/18/1/11.html 20/10/2015
Conclusions

5.1 We live in an increasingly complex world. Systems and models that need to be analyzed are becoming more and more complex. In this context, agent technology could be a powerful and useful tool for researchers. Indeed, programming multiagent systems is rapidly turning into a new discipline of its own. In this article, we have presented all available agent platforms that are or can be used in simulating real-life study cases. This study analyzed twenty four agent platforms developed by different groups, academic or industrial oriented. We figured out that JADE still remains the most popular platform since it is purely designed in Java and supports different kinds of systems operating in the web. Among the platforms are some commercial products which indicate that industry is turning to agent technology. That finding is really hopeful since agent technology has potentials.

5.2 The qualitative comparison and classification intended to provide an overview of the different alternatives. The interested researchers, whenever they need to take a decision upon platform selection, can study the above presented tables (sections 3 and 4) in order to find the intersection that better describes their needs. They can, for instance, start from the domain (Table 9), limiting their options to a platform group, then they can exclude platforms by examine further characteristics such as the programming language (Table 7), the learnability (Table 3) or the current status (Table 2) of the platforms, ending up to the best for them choice.

5.3 We believe that all the platforms compared in the paper are interesting, and that the above comparison could help a developer to decide which platform suites his/her needs better. Such a survey can attract many potential researchers or developers who want to incorporate agent-based technologies into their applications. With this study, we expect to contribute to cover the need of an updated review and encourage future work in the field.

References

ACRONYMICS INC. (2004). AGENTBUILDER: An Integrated Toolkit for Constructing Intelligent. Software Agents. Reference Manual. <http://www.agentbuilder.com>. Archived at <http://www.webcitation.org/6QWerbbZQ>.

Agent Factory (2014). Available at <http://sourceforge.net/projects/agentfactory>. Archived at <http://www.webcitation.org/6QWewhYJw>.

AgentBuilder (2014). Available at <http://www.agentbuilder.com>. Archived at <http://www.webcitation.org/6QWerbbZQ>.

AgentScape (2014). Available at <http://www.agentscape.org>. Archived at <http://www.webcitation.org/6QWf2b9bH>.

AGLOBE (2014). Available at <http://agents.felk.cvut.cz/projects/aglobe>. Archived at <http://www.webcitation.org/6QWf4ybnz>.

AnyLogic (2014). Available at <http://www.anylogic.com/>. Archived at <http://www.webcitation.org/6QWfCrQ IW>.

BELLIFEMINE, F., Caire, G., Poggi, A. & Rimassa, G. (2003). JADE: A white Paper. EXP in search of innovation, 3(3), 6–19.

BORDINI, R., Braubach, L., Dastani, M., Seghrouchni, A., Gomez-Sanz, J., Leite, J., O'Hare, G., Pokahr, A. & Ricci, A. (2006). A survey of Programming Languages and Platforms for Multi-Agent Systems. Informatica, 30, 33–44.

BORDINI, R., Hübner, J. & Wooldridge, M. (2007). Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley Series in Agent Technology. ISBN: 978-0-470-02900-8.

BORSCHCHEV, A. (2013). The Big Book of Simulation Modeling: Multimethod Modeling with Anylogic 6. AnyLogic North America. ISBN: 978-0-9895731-7-7.

BRAUBACH, L. & Pokahr, A. (2013). The Jadex Project: Simulation. Multiagent Systems and Applications, 45, 107–128. [doi:10.1007/978-3-642-33323-1_5]

BRAUBACH, L., Pokahr, A. & Lamersdorf, W. (2008). A Universal Criteria Catalog for Evaluation of Heterogeneous Agent Development Artifacts. From Agent Theory to Agent Implementation (AT2AI-6), 19–28.

BRAUBACH, L., Pokahr, A. & Lamersdorf, W. (2014) Negotiation-Based Patient Scheduling in Hospitals - Reengineering Message-Based Interactions with Services. Advanced Intelligent Computational Technologies and Decision Support Systems 486, 107–121. [doi:10.1007/978-3-319-00467-9_10]

Cormas (2014). Available at <http://cormas.cirad.fr>. Archived at <http://www.webcitation.org/6QWINMG4t>.

Cougaar (2014). Available at <http://cougaar.org/wp/downloads/>. Archived at <http://www.webcitation.org/6QWfQXkN>.

CYBELEPRO (2006). CybelePro Agent Infrastructure, User's Guide v3.0. Distributed Intelligent Systems, Intelligent Automation Inc. <http://www.CybelePro.com>. Archived at <http://www.webcitation.org/6QWfUkZHh>.

CybelePro (2014). Available at <http://www.i-a-i.com>. Archived at <http://www.webcitation.org/6QWfaB5od>.

http://jasss.soc.surrey.ac.uk/18/1/11.html 20/10/2015
LE PAGE, C. & Bousquet, F. (2007). The Cormas platform. Tutorial. ESSA 2007: 4e Conférence de l'Association Européenne de Simulation Sociale. Toulouse, France.

LUKE, S., Cioffi-Revilla, C., Panait, L., Sullivan, K. & Baien, G. (2005). MASON: A Multi-Agent Simulation Environment. In Simulation: Transactions of the society for Modeling and Simulation International, 82(7), 517–527. [doi:10.1177/037549705058073]

MaDKit (2014). Available at <http://www.madkit.net>. Archived at <http://www.webcitation.org/6QWg6q1Fj>.

MASON (2014). Available at <http://cs.gmu.edu/~eclab/projects/mason/>. Archived at <http://www.webcitation.org/6QWg89lRa>.

MILLS D (2009). Industry Roadmap to Web 3.0 & Multibillion Dollar Market Opportunities. Project10X Semantic Wave Report, Washington, DC.

NetLogo (2014). Available at <http://ccl.northwestern.edu/>. Archived at <http://www.webcitation.org/6QWgCXyHq>.

NGUYEN, G., Dang, T.T., Hluchy, L., Laclavík, M., Balogh, Z. & Budinska, I. (2002). Agent Platform Evaluation and Comparison. Institute of Informatics, Slovak Academy of Sciences.

NIKOLAI, C. & Madey, G. (2009). Tools of Trade: A Survey of Various Agent Based Modeling Platforms, Journal of Artificial Societies and Social Simulation, 12(2):2 <http://jasss.soc.surrey.ac.uk/12/2/2.html>.

NORTH, M.J., Howe, T.R., Collier, N.T. & Vos, J.R. (2007). A Declarative Model Assembly Infrastructure for Verification and Validation. In Advancing Social Simulation: The First World Congress (129–140). Springer, Heidelberg.

OEY, M.A., van Splunter, S., Ogston, E., Warnier, M. & Brazier, F.M.T. (2010). A Framework for Developing Agent-Based Distributed Applications. Proceedings of the 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-10), IEEE/WIC/ACM, 2, 470–474. [doi:10.1109/WI-IAT.2010.134]

Repast (2014). Available at <http://repast.sourceforge.net/>. Archived at <http://www.webcitation.org/6QWgJsJfv>.

RESNICK, P., Kuwabara, K., Zeckhauser, R. & Friedman, E. (2000). Reputation systems. Communications of the ACM, 43(12), 45–48. [doi:10.1145/355112.355122]

RUSSELL, S., Jordan, H., O'Hare, G.M.P. & Collier, R.W. (2011). Agent Factory: A Framework for Prototyping Logic-Based AOP Languages. In Proceedings of the Ninth German Conference on Multi-Agent System Technologies (MATES 2011), 6973, 125–136. [doi:10.1007/978-3-642-24603-6_13]

SCHURR, N., Marecki, J., Tambe, M., Scerri, P., Kasinadhuni, N. & Lewis, J.P. (2005). The Future of Disaster Response: Humans Working with Multiagent Teams using DEFACTO (PDF). In AAAI Spring Symposium on AI Technologies for Homeland Security.

SeSAm (2014). Available at <http://www.simsesam.de>. Archived at <http://www.webcitation.org/6QWgNBeCL>.

SISLAK, D., Rehak, M., Pechoucek, M. & Pavlicek, D. (2006). Deployment of A-globe Multi-Agent Platform. In Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 1447–1448. [doi:10.1145/1160633.1160908]

SONNENSSA, M. (2004). JAS: Java Agent-based Simulation library. User's Guide version 1.0. <http://jaslibrary.sourceforge.net/>.

SUN, R. & Naveh, I., (2004). Simulating Organizational Decision-Making Using a Cognitively Realistic Agent Model. Journal of Artificial Societies and Social Simulation, 7(3), 5 <http://jasss.soc.surrey.ac.uk/7/3/5.html>.

Swarm (2014). Available at <http://www.swarm.org>. Archived at <http://www.webcitation.org/6QWgU1rp>.

SWARMFEST (2012). SDG's annual agent-based modeling conference. <http://www.swarm.org/index.php/Swarmfest_2012>. Archived at <http://www.webcitation.org/6QWgWxctp>.

TISUE, S. & Wiensky, U. (2004, updated 2013). NetLogo: Design and implementation of a multi-agent modeling environment In Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence, Chicago, Illinois.

WINIKOFF, M. (2005). JACK intelligent agents: An industrial strength platform. In Multi-Agent Programming. Springer, 15, 175–193.