AGENT TOOLKITS: A GENERAL OVERVIEW OF THE MARKET AND AN ASSESSMENT OF INSTRUCTOR SATISFACTION WITH UTILIZING TOOLKITS IN THE CLASSROOM

By

Alexander Serenko and **Brian Detlor**

Michael G. DeGroote School of Business
McMaster University
Hamilton, Ontario

Working Paper #455

July, 2002

ABSTRACT

This report presents results from a recent research investigation on agent toolkits on the market and their use in post-secondary courses. Data collection comprised the download and trial use of 20 agent toolkits, as well as the completion of an online questionnaire on instructor satisfaction by 87 post-secondary course instructors from around the world. The report identifies four major categories of agent toolkits on the market. It appears that no single uniform toolkit satisfies the needs of all agent-related courses. Research findings also suggest that satisfaction levels are influenced primarily by user interaction with the toolkit, followed to a lesser extent by toolkit performance and functionality. The report identifies features an ideal agent toolkit should possess and concludes with ideas for future research.

KEYWORDS

agent toolkits, intelligent agents, post-secondary education, instructor satisfaction

ACKNOWLEDGEMENTS

This paper is partially supported by a grant from the Natural Sciences and Engineering Council of Canada.

TABLE OF CONTENTS

1. Introduction 1

1.1. Purpose of the Project 1

1.2. Why needed? 2

2. Background 3

2.1. Intelligent agents 3

2.1.1. What is an agent? 3

2.1.2. Major Problems of Agent Building 4

2.2. Agent Toolkits 5

2.2.1. What are Agent Toolkits? 5

2.2.2. Why are Agent Toolkits needed? 6

3. Agent Toolkits on the Market 6

3.1. Mobile Agent Toolkits 9

3.2. Multi-Agent Toolkits 11

3.3. General Purpose Agent Toolkits 13

3.4. Internet Agent Toolkits 16

3.5. Agent Toolkit Manufacturers 18

4. Satisfaction of Course Instructors with Agent Toolkits 19

4.1. Methodology 19

4.1.1. Questionnaire 19

4.1.2. Participant Recruitment 22

4.2. Results 23

4.2.1. Instructors Who Utilized Agent Toolkits 23

4.2.2. Instructors Who Did Not Utilize Agent Toolkits 35

5. Discussion and Conclusions 38

6. References 42

Working Papers – Recent Releases 44

1. Introduction

In today's brave new Internet world, a newer form of software, called *intelligent agents*, offers people the potential to navigate and utilize Web-based information resources more effectively and efficiently than ever before. Intelligent agents are software programs that act on behalf of users to find and filter information, negotiate for services, automate complex tasks, and collaborate with other agents to solve complex problems (AgentBuilder, 2000). Agents perform these tasks continuously and autonomously in particular environments often inhabited by other agents and processes (Shoham, 1997, pp. 271-72).

The use of agents has been well-documented in the electronic commerce domain (Maes, 2001; 1994; Maes, Guttman & Moukas, 1999; Rahman & Bignall, 2001), especially in terms of industrial, commercial, medical, and entertainment applications (Jennings & Wooldridge, 1998). With the advent of the Semantic Web proposed by Tim Berners-Lee, agents are envisioned to play a more significant role in the near future (Port, 2002; Berners-Lee, Hendler & Lassila, 2001). Given the importance and rise of this newer form of software, agent toolkits are becoming more of a necessity to help build, re-use, and deploy intelligent agents. As such, agent toolkits are being introduced and incorporated in the curriculums of post-secondary education courses geared to train the next generation of electronic commerce managers and programmers.

To gain insight on the current landscape of agent toolkits available on the market and their use in post-secondary education courses, a project was conducted by the authors from June 2001 to April 2002. Data collection and analysis involved the download and trial use of 20 agent toolkits, as well as the recruitment of 87 post-secondary course instructors across the globe to answer a Web-based questionnaire concerning instructor satisfaction with agent toolkit use in the classroom. This paper reports on the project's findings.

1.1. Purpose of the Project

The purpose of the project was twofold. The first was to conduct a general comparison of agent toolkits on the market in terms of the functionality they offer and their underlying technology. Although agent toolkits have had a relatively short history on the software market so far, there are many toolkits now available. They differ in terms of the functionality they offer, their ease of

use, area of application, and underlying technology. The goal was to provide a high-level overview and assessment of these varying toolkits.

The second, and more important purpose, was to evaluate the use and satisfaction of course instructors utilizing agent toolkits in their curriculums. The authors wanted to assess course instructor satisfaction utilizing different agent toolkits across various intelligent agent-based courses, in the hopes of making recommendations on the use of particular toolkits in post-secondary education.

As such, the project addressed the following three research questions:

1) What are the available agent toolkits in the market and how do they compare in terms of their general functionality and underlying technology?

2) How are agent toolkits utilized in post-secondary courses?

3) How satisfied are post-secondary course instructors with utilizing agent toolkits in their courses?

1.2. Why needed?

There are several reasons why answers to these research questions are needed. First, intelligent agents are becoming extremely popular in Internet and high-tech industries. It can be assumed that in a few years all Web browsers will have a built-in personal assistant, travel, or purchasing agent and that people will feel comfortable using these tools. An overview of agent toolkits on the market will help agent developers and educators become more aware of the choices available at their disposal.

Second, more post-secondary schools are offering graduate and undergraduate courses in intelligent agents where students study agent technologies by experimenting with existing agents or building new ones. There is a lack of research that examines agent toolkit utilization in post-secondary education and the success instructors have had with them. An examination of instructor satisfaction with agent toolkit use in the classroom is warranted.

Last, despite the wide variety of agent toolkits available on the market, there is no comprehensive, one-stop list that evaluates these toolkits. This report will help bridge that void.

2. Background

2.1. Intelligent agents

For thousands of years, people have always thought about someone doing tasks for them. It could be a robot or personal assistant who may not only perform tasks but also predict people's needs and preferences. Moreover, the assistant could think the same way the owner does. This is the potential of agent technology. Unfortunately, not until the end of the twentieth century has it been possible to create such software. Now, with the rapid growth of the computer industry and telecommunication networks, this dream is becoming reality with recent advancements in agent technology.

2.1.1. What is an agent?

An "agent" is basically someone who acts on someone else's behalf. Usually, the agent refers to a human. For example, the American Heritage Dictionary defines an agent as "one that acts or has the power or authority to act… or represent another." In the computer world, an agent most often refers to a software program that acts on a user's behalf. For example, the agent collects and analyzes information, draws conclusions, makes recommendations, and performs transactions. Webopedia[1] defines an agent as a program that performs some information gathering or processing task in the background. On the Web, these tasks typically refer to the retrieval and delivery of information. Maes (1999) defines an electronic commerce agent as a piece of software that profiles users or buyers to provide personalized service.

All agents are not created equal. Some are more advanced than others. Basic software agents exhibit the common characteristics of autonomy (independence), persistence (long-livedness), monitoring of the environment, and communication and collaboration with other agents and/or

[1] Available at http://www.webopedia.com

the user. More “intelligent” agents possess higher-level abilities, such as mobility, decision-making, and the ability to learn. According to Sinmao (1999), the continuum of intelligent agents may be characterized along three dimensions: agency, intelligence, and mobility.

Agency reflects the degree of independency of an agent. The agent should be able to collect and analyze information as well as perform tasks independently of the user using its knowledge about the user’s profile.

Intelligence is the degree of learning by an agent. The agent should be able to perceive, understand and analyze the environment. Given that the environment constantly changes, the agent should be capable of learning and adapting to these changes. Further, the agent should draw conclusions from the information collected and perform actions on the user’s behalf.

Mobility is the last dimension to consider. Although agents may be static (e.g., an e-mail assistant that sorts incoming messages), more intelligent agents require some degree of mobility. The agent should be able to travel from one machine to another gathering information that it will process later. Furthermore, it may perform different tasks on a remote machine on behalf of the user. For example, a travel agent may visit a number of sites, gather information on prices and then process it on the user’s machine or negotiate the details of the deal with other agents in cyberspace and even sign a binding agreement.

2.1.2. Major Problems of Agent Building

According to Maes (1994), there are two main problems that confront the building of software agents. The first is one of competence: how does an agent acquire the knowledge it needs to decide when and how to help the user? It appears that the quality and competence of an agent will depend on the ability of its software developer to implement all desired features. The second is one of trust: how can we guarantee a user feels comfortable delegating tasks to an agent? This second problem is becoming less important nowadays when students and researchers develop their own agents (e.g., personal digital assistants or shopping agents) using agent toolkits. It is likely that user trust will be greater in those instances where people personally create agents, supplying them with knowledge and logic rules, compared to those instances where users utilize

pre-canned, “off-the-shelf” agents. In addition, the level of trust a user has for an agent will depend on the user’s experience with utilizing such toolkits.

2.2. Agent Toolkits

Currently there are a variety of toolkits available on the market ranging from general agent development platforms, like AgentBuilder developed by Reticular Systems, to highly specialized tools, like Excalibur developed by the Technical University of Berlin which allows for the creation of autonomous agents in a complex computer-game environment. The AgentBuilder web site[2] identifies numerous agent toolkits available on the market.

2.2.1. What are Agent Toolkits?

There is no universal definition of agent toolkits. Each vendor uses its own explanation of the term. For example, Reticular Systems states that its AgentBuilder toolkit application “is an integrated tool suite for constructing intelligent software agents” (AgentBuilder, 2000)[3]. Authors of the Java Agent Development Environment (JADE) define their toolkit as “a software framework to make easy the development of agent applications … for interoperable multi-agent systems” (Bellifemine et al., 2000).

For this report, an agent toolkit is defined as any software package, application or development environment that provides agent builders with a sufficient level of abstraction to allow them to implement intelligent agents with desired attributes, features and rules. Some toolkits may offer only a platform for agent development, whereas others may provide features for visual programming.

Agent toolkits may also provide an environment for running, monitoring, analyzing and testing agents, which is very important for both researchers and students learning about agent

[2] See “Agent Construction Tools” at http://www.agentbuilder.com/AgentTools/index.html

[3] See also http://www.agentbuilder.com/Documentation/product.html

technologies. For example, in case of multi-agent systems, an agent development environment provides a context for agent interaction and sets of governing rules.

2.2.2. Why are Agent Toolkits needed?

The reasons why agent developers use agent toolkits is similar to those reasons why software developers who deal with object-oriented programming (OOP) prefer to use special development environments like Java VisualAge or Microsoft Visual Basic. First, they provide a certain level of abstraction in which programmers can develop their objects. Second, they incorporate some features of visual programming, which saves much time and makes development easier, more attractive and enjoyable. Third, they offer run-time testing and debugging environments. Finally, they allow programmers to reuse classes (definitions of objects) created by other programmers.

Unfortunately, existing OOP development platforms and compilers do not support all facets of agent development. For example, they do not address the implementation of agent features, agent interaction rules, communication language, and a common knowledge base. This is why a new suite of agent toolkits has appeared on the market in the last few years: to create a development environment that fully supports agent creation, testing, and reuse.

3. Agent Toolkits on the Market

To assess the available agent toolkits on the market, the authors utilized the AgentBuilder web site list as a starting point, as it was posted on July 2001. This list was amended by the authors' own secondary research on the Web to find other agent toolkits available on the market. From this investigation, 40 agent toolkits were downloaded from their vendor's web sites; 20 were successfully installed locally and individually tested. Installing the software was a time intensive process given the strict configuration requirements of some of the toolkits. For example, many toolkits worked only with a certain version of Java Virtual Machine (JVM) that sometimes required uninstalling and installing previous versions of JVM.

Each of the toolkits was assessed in terms of its underlying technology and general functionality. Overall, four major categories of agent toolkits were identified: mobile agent toolkits, multi-agent toolkits, general-purpose toolkits and Internet agent toolkits. Table 1 below provides a

summary of the authors' assessment of the more popular agent toolkits in terms of these major categories.

	Features	**Technology**	**Examples**	**Comments**
Mobile agent toolkits	Mobility Communication (desirable)	Java (77%) Python(7%) Tcl(7%) C/C++(6%)	Concordia Gossip FarGo	Addresses both mobility and communication.
			IBM Aglets	A tool for manipulating mobile agents. Knowledge of Java required.
Multi-agent toolkits	Agent Interaction Communication Coordination Conflict Resolution	Java	MadKit	Allows running sample built-in agents as well as agents created outsize the environment.
			Zeus	A convenient tool for developers who lack Java programming. May be used in basic agent courses.
			JADE	Offers tools for creating and debugging MAS.
			JATLite	Well-addressed communication.
			MAST	Knowledge interchange.
General purpose toolkits	No specific area of concentration No unique requirements	Java(68%) Prolog(8%) C/C++(8%) Other(16%)	FIPA-OS	Best choice for development of FIPA compliant agents. Knowledge of Java is required for advanced development.
			Ascape	May be used by programmers and non-programmers.
Internet agent toolkits	Have features common to mobile agents Usually interactive and emphasize personalization	Java	Microsoft Agent	No programming skills required.
			Voyager	Allows creating mobile internet agents. Knowledge of Java required for development.
			NetStepper	Used for creating information retrieval agents. No programming skills required.

Table 1: Overview of agent toolkits on the market

Mobile agent toolkits were defined to be toolkits primarily dedicated to the creation of mobile agents. A mobile agent is an executing program that can migrate, at times of its own choosing, from machine to machine in a heterogeneous network. On each machine, the agent interacts with stationary service agents and other resources to accomplish its tasks (Gray et al., 2000). In other words, it is an object that can move from one computer or host machine to another performing a set of tasks that are specified by a user. A remote super computer that becomes a "place of agent

meeting" for information exchange may act as a broker linking together agents performing similar tasks.

M*ulti-agent toolkits* were defined to be toolkits that were primarily concerned with the development of a multi-agent system (MAS). A MAS is usually composed of several interacting agents. This interaction involves the coordination of actions between agents, and the adaptation of agent behavior in response to the environment. MAS is an emerging area of research in the field of distributed artificial intelligence. Often, an agent cannot solve a complex problem alone and needs cooperation with other agents to exchange data and information or delegate tasks. Over the last five years, there has been rapid development and deployment of MAS. Many conferences, workshops and seminars have been organized around this topic. There are also several web sites (e.g., www.multiagent.com) that are entirely devoted to MAS. DeLoach (2001) identifies numerous challenges confronting successful MAS building. These include: decomposing problems and allocating tasks to individual agents; coordinating agent control and communications; making multiple agents act in a coherent manner; reasoning about other agents and the state of coordination; reconciling conflicting goals between the agents; and engineering practical multi-agent systems.

General purpose agent toolkits were defined to be agent development environments that do not concentrate on one specific area of agent development, like the toolkits discussed above. Instead, they allow users to create different kinds of agents for different purposes. Toolkits in this category are also identified by vendors as being agent development environments, agent development tools, agent frameworks, or agent architectures.

Internet agent toolkits were defined to be toolkits dedicated to the creation of Internet agents. These agents are dedicated to searching and retrieving information off the Internet and/or improving the interface with which users themselves search the Web. Various functions may be performed by Internet agents. They may search the Web while a user is not surfing. They may automate routine and time-consuming tasks (e.g., sorting the results that come back from search engines). They may find products a user needs or make the surfing experience more interesting and pleasant. Internet agents are usually interactive and emphasize personalization of retrieved search results.

The following subsections provide more detail on the agent toolkits that were analyzed. Each subsection pertains to one of the four major categories of agent toolkits identified above.

3.1. Mobile Agent Toolkits

In terms of mobile agent toolkits, the authors found the Mobile Agent List[4] by Fritz Holh to provide a very comprehensive list of mobile agent construction tools. The list offers a description of 60 products. Although these products differ in terms of functionality, technology and area of application, all of them share two common features: mobility and communication. Mobility is necessary for all mobile agents since they must move through communication networks. For example, Concordia[5] implements mobility through its queue manager for the reliable transport of agents and has a special Java object serialization scheme to accomplish this. Communication is another important feature. Although optional, agents may need to communicate and exchange messages while traveling in cyberspace. Some agent toolkits support the creation of mobile agents which are capable of communicating with one another. For example, IBM supports aglets[6], which is a mobile agent construction tool with message passage features: agents create and pass both asynchronous and synchronous message objects to one another to communicate.

Figure 1 illustrates the underlying technology employed in mobile agent toolkits. The diagram shows how Java, due to its unique platform independence, has been the most popular language for development of mobile agent toolkits. The other languages utilized for development are Python (7%), Tcl (7%), C/C++ (6%), Prolog, Perl and Jess (3%).

---

[4]Available at http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/preview/preview.html

[5]Developed by Horizon Systems Laboratory, Mitsubishi Electric Technology Center. Available at http://www.concordiaagents.com

[6] Available at http://www.trl.ibm.com/aglets

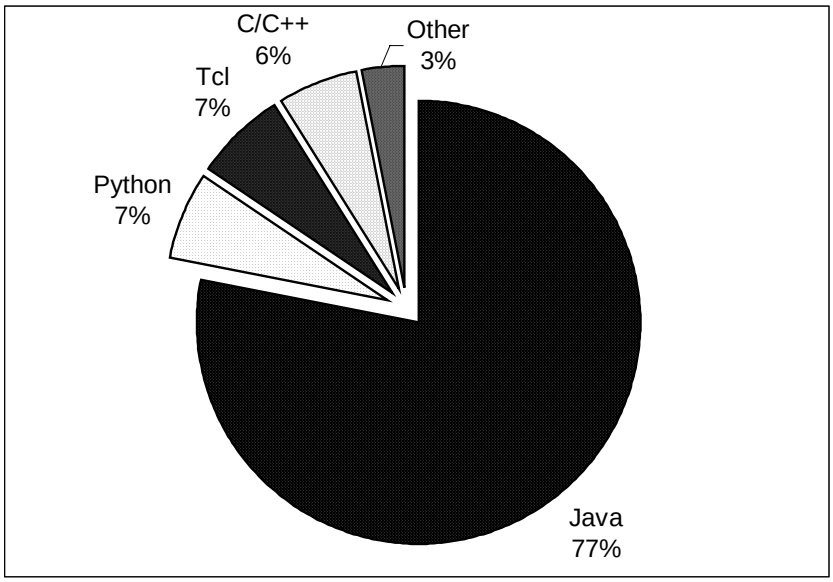

Figure 1: Underlying technology of mobile agent toolkits

Other examples of mobile agent toolkits are Gossip, developed by Tryllian[7], which allows the user to build an agent that trades information on his or her behalf over the Internet, and FarGo, developed by Technion[8], which facilitates the creation of agents with unique relocating control features.

Aglets, as mentioned above, is a mobile-agent development tool. The Aglets Software Development Kit[9] is a development environment for mobile Internet agents called aglets. The aglet represents a Java object that may be transported from one host machine to another together with state information. The aglet may work on a certain host computer and then it may stop execution and move further though the Internet keeping the data it has gathered and processed. After arriving at a new point of destination, the aglet resumes working. The aglet development kit utilizes J-AAPI technology – a standard for interfacing aglets and their environment, which defines the set of behaviors aglets follow, such as: message creation and handling, initialization,

[7] See http://www.tryllian.com

[8] Technion is developed by the Israel Institute of Technology. See http://www.dsg.technion.ac.il/fargo/about.php3

[9] Developed at the IBM Tokyo Research Laboratory. See http://www.trl.ibm.com/aglets/about.html

dispatching, retraction, activation and deactivation, cloning and disposing (Lange, 2002). J-AAPI, a language created for aglet programmers, can also be used for educational purposes.

The toolkit has a user-friendly development environment, called Tahiti, that provides a graphical user interface with a variety of options, for example, configuration options and building tools. Tahiti includes a number of sample aglets that may be run and analyzed by first-time users, which is another important feature of the package. Note, however, that aglets are Java classes that should be created outside Tahiti. The toolkit does not provide a Java development environment itself. Rather, it allows users to send aglets into cyberspace, and monitor and manage them. Therefore, the user must be able to create his or her own agents using an Integrated Development Environment (IDE) like Symantec's Visual Café, IBM's VisualAge, Borland's JBuilder or Metrowerk's Code Warrior. The authors found IBM's aglets to be a useful tool for manipulating mobile agents, however, it requires the agent developer to be very familiar with Java programming. As such, instructors wishing to utilize aglets in educational courses or research must be strongly acquainted with Java.

3.2. Multi-Agent Toolkits

With respect to multi-agent toolkits, the authors discovered that several different toolkits are available. For example, out of the 60 agent toolkits presented on the AgentBuilder web site, eight were multi-agent. All these utilize Java for agent development. An example of a multi-agent toolkit is the Java Agent Development Framework (JADE)[10], developed by the University of Parma, which simplifies implementation of multi-agent systems and provides a set of tools that support development and debugging. The Java Agent Template Lite (JATLite)[11] is another example multi-agent toolkit. Created by Stanford University, JATLite allows users to build agents that communicate robustly over the Internet though the Agent Message Router Facilitator.

[10] JADE is developed by the University of Parma. See http://sharon.cselt.it/projects/jade

[11] See http://java.stanford.edu/java_agent/html

Developers of the Multi-agent Systems Tool (MAST)[12] at the Technical University of Madrid also emphasize all aspects of knowledge interchange among agents.

A well-known multi-agent toolkit is Zeus[13], developed by the British Telecommunications Laboratory. This toolkit offers a library of software components and tools that facilitate fast and friendly design, development, and deployment of multi-agents. Zeus consists of three main functional components: the agent component library, agent building tools, and visualization tools. The agent component library is a collection of software components that implement multi-agent functionality. It offers a number of pre-written, standard agent components and a collection of sample agents for developers to utilized. The agent building tools is an integrated multi-agent development environment for creating agents. The visualization tools is a runtime environment for running, testing and debugging agents. The multi-perspective approach used in the visualization tools area gives developers an opportunity to visualize the processes they select.

According to the documentation available off the Zeus home page, creating Zeus agents involves the following five consecutive steps: 1) ontology creation; 2) agent creation; 3) utility agent configuration; 4) task agent configuration; and 5) agent implementation. For the first step, users define their agents' ontology – the declarative knowledge representing concepts, attributes and values. This is done via an ontology editor. The second step is where agent creation occurs. This is accomplished through an agent definition user interface where agent tasks and coordination are described. In the third step, the user creates utility or facilitator agents that collectively provide an working infrastructure to support agent task fulfillment and collaboration. In the fourth step, the designer specifies the hosts on which the agents will run and external programs to which the agents will be linked. The last step involves the generation of agent source code. This is done through a code generator which creates a Java-based agent application as its output as well as a number of utilities and batch files. Once an agent has been compiled, the developer may move it to a host machine for testing and running using a variety of graphical, report and statistical tools. It is the authors' opinion that Zeus is a very convenient educational tool for

[12] See http://www.gsi.dit.upm.es/~mast

[13] The Zeus homepage is available at http://193.113.209.147/projects/agents.htm

developers who lack a strong Java programming background. This toolkit may be used by instructors in basic intelligent agent courses.

MadKit[14] is another multi-agent toolkit for consideration. Developed by the MadKit Development Group, MadKit is a Java multi-agent platform built upon an organizational model. It provides general necessary agent facilities and allows heterogeneity in agent architectures and communication languages as well as customization. G-Box is the graphical user interface development environment for MadKit. The major components of G-Box are a toolbox that shows a list of available agents, a properties zone that shows an agent's editable properties, and an agent desktop working environment. In addition, there are special components within G-Box which show lists of currently running agents and groups of agents.

G-Box offers a wide range of sample multi-agents for execution, as well as the ability for users to add their own Java-written agent applications. This functionality may be of specific interest to educators. For example, to teach basic functionality of multi-agents, an instructor may ask students to monitor the sample Ping-Pong agent available within MadKit. After launching, this agent creates a "Ping-Pong group" and waits for another agent to play with. Later, students can launch a second agent which registers with the "Ping-Pong" group. This is detected by the first agent, and the two agents begin interacting with one another by exchanging messages. "Mosquitoes" is another interesting example of multi-agent simulation of potential interest to students where small agents are created according to a predefined algorithm. To view and test out these agents in action, MadKit offers a Java-enabled browser. Such visualization tools can help promote greater learning and understanding of basic agent operability.

3.3. General Purpose Agent Toolkits

In terms of general purpose agent toolkits, a larger majority of toolkits fall into this category. For example, general purpose agent toolkits constituted over 40% of all applications presented on the AgentBuilder web site. Figure 2 illustrates the underlying technology employed in this category of toolkits.

---

[14] Refer to http://www.madkit.org

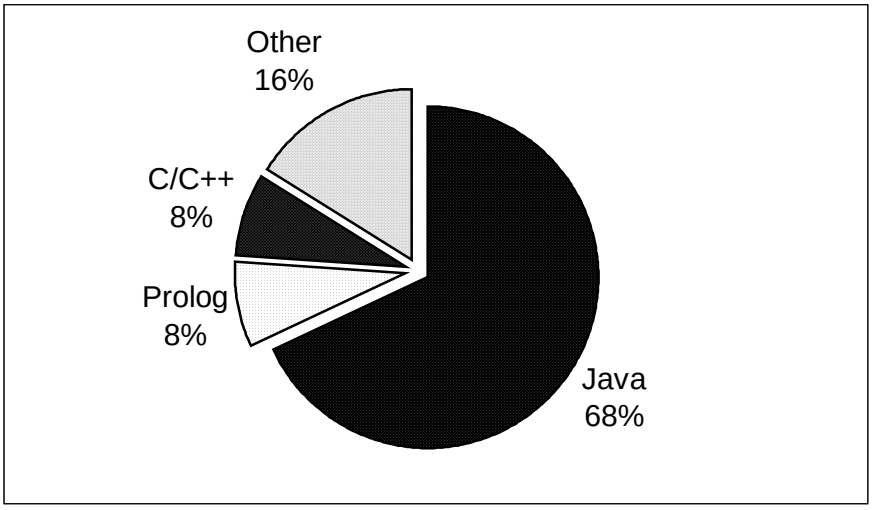

Figure 2: Underlying technology of general purpose toolkits

Java, again, is the leading programming language. Sixteen percent of the applications have been built in C/C++ and Prolog, and a few toolkits have been created with ActiveX, Jack, COOL, Perl, VisualWorks or Lisp.

FIPA-OS and Ascape are two examples of general purpose agent toolkits. FIPA-OS[15], developed by Emorphia Limited, is a toolkit that develops FIPA[16] compliant agents. FIPA (the Foundation for Intelligent Physical Agents) is an international organization dedicated towards creating and implementing software standards for heterogeneous and interacting agents and agent-based systems. Standards developed by FIPA aim to promote the interoperation of heterogeneous agents and the service that they can represent. FIPA-OS is a component-based agent toolkit that allows users to develop FIPA compliant agents using three groups of components: 1) mandatory components, which must be included in all FIPA compliant products; 2) components with switchable implementations; and 3) optional components, such as Database Factory, Parser Factory and Choice Constraint Language.

Figure 3 below presents the graphical user interface of the toolkit.

---

[15] See http://fipa-os.sourceforge.net

[16] See http://www.fipa.org

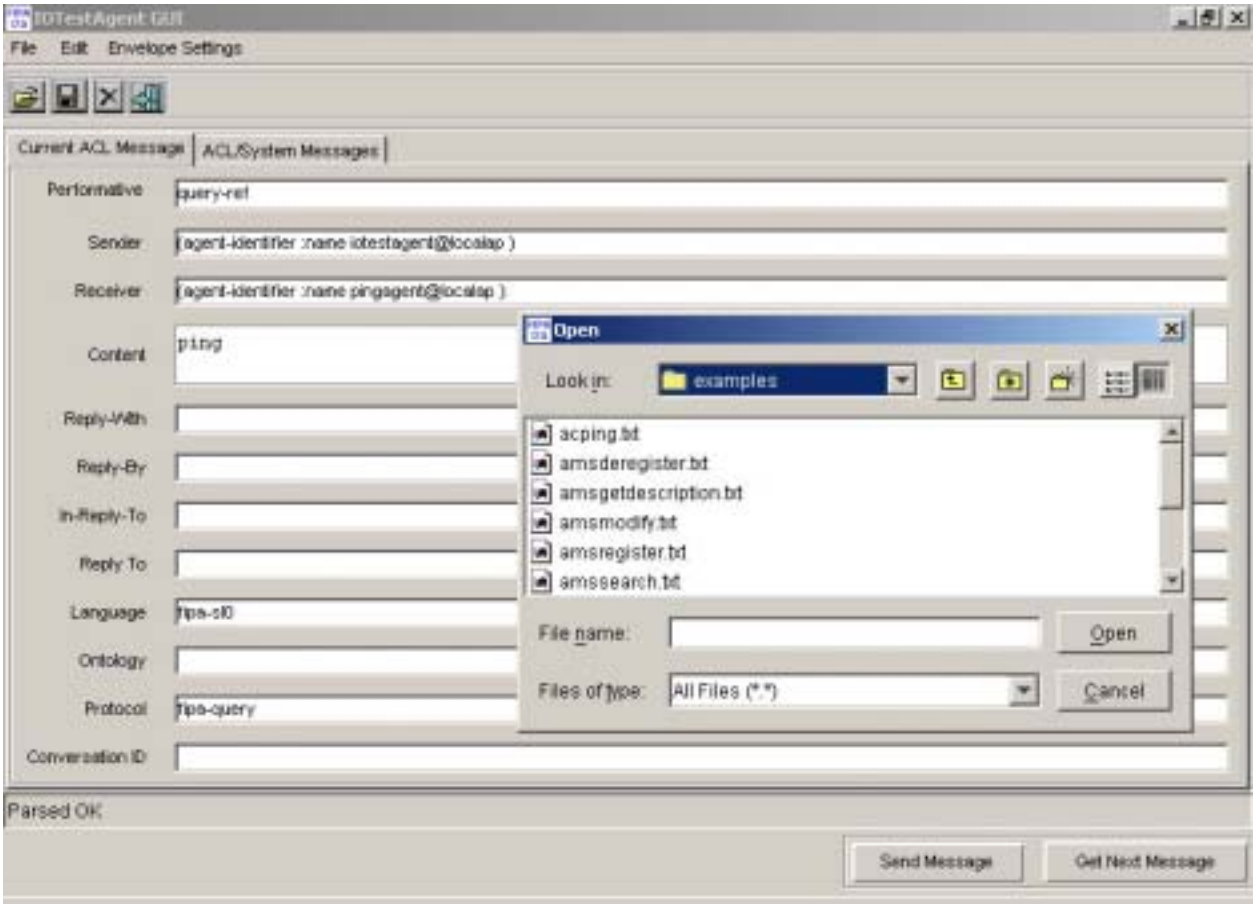

Figure 3: The FIPA-OS Agent Toolkit (kindly reproduced with permission)

FIPA-OS is built utilizing Java technology. Agent developers utilizing this toolkit are required to know the concepts of object-oriented programming and be familiar with Java classes. The toolkit provides a group of classes that are used in agent development, as well as a graphical testing environment. The simple tutorials available on the FIPA-OS web site can be used by course instructors to teach students the basic workings of agent development. For example, using these tutorials, students may easily create the following agents:

- A *generic agent* which is the basic agent and may be used in further development of more complicated agents;
- A *search agent* which is based on the generic agent;
- A *ping agent* which allows the builder to understand reactive behavior within the agent;

- A *JESS agent* which utilizes the JESS engine (an expert system shell) to help developers write their own intelligent agents; and

- *eMarkets* which is an agent based simulation environment for an electronic marketplace.

Performing these exercises will enhance students' understanding of agents and give them some practical experience in constructing FIPA compliant intelligent agents.

Ascape[17], introduced by the Brookings Institution, is a software framework for developing, testing, and analyzing intelligent agent-based models. In this environment, agent objects exist within "scapes". These are agents that represent collections (arrays and lattices) of agents and provide interaction and behavioral rules.

Ascape is written purely in Java and may run on any platform. The toolkit, however, may be used by both programmers and non-programmers alike. The toolkit offers a graphical interface where developers who do not have familiarity with Java may monitor and manage their own agents. For example, users have complete control over model parameters at runtime. Users may also create graphs and customize their own views. In future versions of Ascape, no programming will be required to change basic rules and structures. Ascape supplies itself with ten sample demonstration agents (e.g., Prisoner's Dilemma, Retirement, Class Emergence, Firms, Sugarscape and Norms). Students of agent technology may find these samples helpful. To create robust agents in Ascape requires knowledge of Java. However, development of simple agents requires only basic knowledge of Java and object-oriented programming. This is in part due to Ascape's ability to allow the construction of simple agent models from "off-the-shelf" parts. More advanced developers would likely build their agents utilizing Java classes supplied with the agent.

3.4. Internet Agent Toolkits

With respect to Internet agent toolkits, the authors found several toolkits for consideration. The authors would like to point out three in particular: Microsoft Agent, Voyager, and NetStepper.

The Microsoft Agent toolkit supports the creation of a personal interactive animated character that may be hosted on a user's web site. This agent may guide visitors through the user's site, explain menus, navigational tools, offer tips and even talk. A benefit for students of agent technologies is that the Microsoft Agent toolkit requires no knowledge of programming to generate these interface agents.

Voyager[18], designed by ObjectSpace Inc, is a toolkit for building Java-based agent brokers used in agent-mediated electronic commerce. The main objective of the toolkit is to ease the design, development and deployment of distributed applications over networks such as the Internet. The idea is to build autonomous agents that act on behalf of the client, are able to move through the network, and which are in constant communication and collaboration with other agents and the user. The Voyager product line includes the following applications:

- a Voyager Object Request Broker (ORB) that automatically supports communication among distributed component object models (DCOM);
- a Voyager ORB Professional, which inherits all Voyager ORB features and adds a graphical user interface and a few new features;
- Voyager Security - a security framework for implementing security features such as secure socket layer and firewall tunneling;
- Voyager Transactions for distributed transactions support; and,
- a Voyager Application Server that helps handle computing needs on the enterprise level.

In terms of classroom use, Voyager facilitates the construction of agents that can be viewed by users within a development console. However, Voyager requires thorough knowledge of Java to develop and deploy agents within the toolkit.

---

[17] Refer to http://www.brook.edu/es/dynamics/models/ascape

[18] Refer to http://www.objectspace.com

NetStepper[19] is another tool for designing, testing and running Internet agents that collect information off the Web. The major advantage point of NetStepper is that it allows all aspects of agent design to be accomplished through its graphical user interface. This feature makes the toolkit very attractive to non-programmers. To create an agent within NetStepper, a user can employ a wizard to help describe the agent's functionality. Working with the toolkit, users can build their own databases of the Internet where all information is represented by links. The application includes a very easy to follow tutorial allowing novices agent developers to create simple agents within a few minutes. After an agent has been created, the user may run it from within the toolkit. The agent's output is represented either as a list of the links pointing to the required information, or as a text file containing these links.

3.5. Agent Toolkit Manufacturers

There are a number of different manufacturers of agent toolkits on the market. The manufacturers of agent toolkits presented on the AgentBuilder web site as of July 1, 2001 can be divided into three general categories: academic, commercial, and non-profit developers.

The first and largest group, which has created over half of all agent development tools, are *academic manufacturers* represented by colleges, institutes, universities and their laboratories. Usually, their products, supporting documentation, and research papers are available on their web sites free of charge.

The second largest group are *commercial manufacturers* represented by commercial and profit-oriented companies that charge anywhere from $100 to $2,000 (US) per license.

The smallest group are *non-profit manufacturers*, such as research consortiums and government-supported organizations.

Figure 4 below presents the breakdown of agent toolkits developers by group.

[19] See http://www.internet-search-agents.com/netstepper

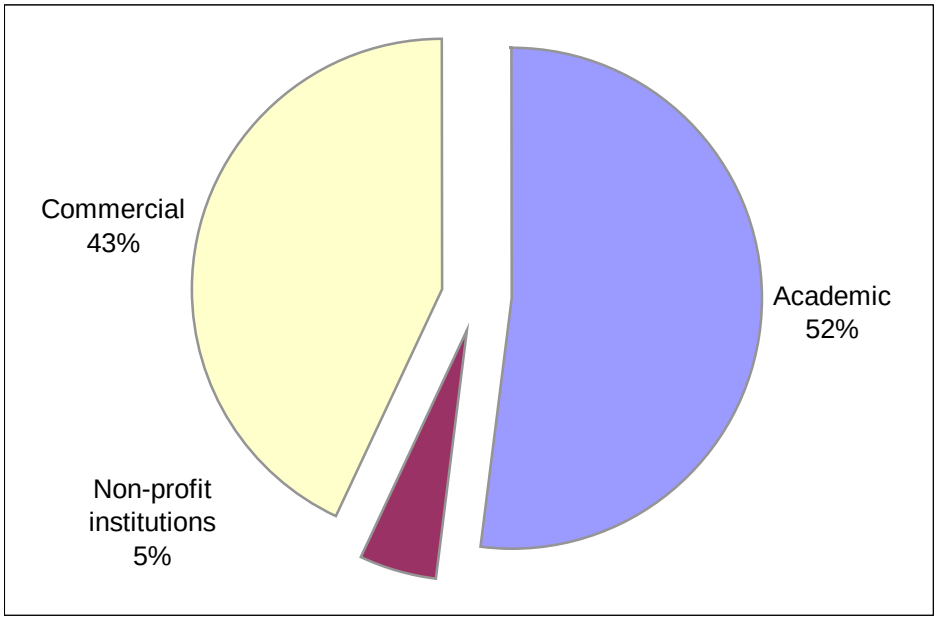

Figure 4: Developers of agent toolkits

4. Satisfaction of Course Instructors with Agent Toolkits

The previous section of this report assessed the available agent toolkits on the market. Attention now turns to the authors' findings on the use and satisfaction of course instructors utilizing agent toolkits in their curriculums.

4.1. Methodology

4.1.1. Questionnaire

The first step in obtaining sufficient information to assess post-secondary instructor satisfaction with agent toolkits was to design a Web-based questionnaire to send to instructors. The purpose of the questionnaire was to poll instructors' perceptions and levels of satisfaction with utilizing agent toolkits in the classroom. The questionnaire was designed for instructors of agent-related post-secondary courses who currently utilized agent toolkits in the classroom, as well as those who might in the future.

Table 2 below outlines the questions that were asked of instructors who said they did not use or demonstrate an agent toolkit in one of their courses.

Question	Answer	Purpose
1. Why have not you used or demonstrated an agent toolkit in your course?	Free-form text	To provide specific information on the reasons why some instructors did not include agent toolkits as a part of their curriculums.
2. Do you think utilizing or demonstrating an agent toolkit in your course would be beneficial to students?	Yes / No	To help understand instructors' perception of possible benefits of utilizing of agent toolkits in their courses.
3. Please explain why.	Free-form text	To provide insights on instructors' opinions about the usefulness of agent toolkits in post-secondary courses.

Table 2: Questions asked of instructors who did not utilize agent toolkits

Table 3 below outlines the questions that were asked of instructors who said that they used or demonstrated an agent toolkit in one of their courses.

Question	Answer	Purpose
1. What agent toolkits did you use?	Free-form text	To gather background information on agent toolkits to make a distinction among different agent toolkits used by different instructors.
2. List the names of the courses in which you used or demonstrated an agent toolkit	Free-form text	It was believed that different toolkits might be used in different courses and students' performance might depend on their background, which can be inferred from a course name.
3. As an instructor, how satisfied are you with the last agent toolkit you used?	1 - Not Satisfied; 2 - Somewhat Satisfied; 3 - Satisfied; 4 - Very Satisfied	To measure instructor satisfaction with a particular toolkit and discover a relationship between a particular toolkit used in the course and a level of satisfaction of instructors.
4. Explain your reasons	Free-form text	To understand the underlying reasons of the level of instructors' satisfaction identified in the previous question.
5. In your opinion, how satisfied were your students who utilized or saw a demonstration of this agent toolkit?	1 - Not Satisfied; 2 - Somewhat Satisfied; 3 - Satisfied; 4 - Very Satisfied	To measure the level of students' satisfaction as perceived by the instructor.
6. Explain your reasons	Free-form text	To understand the underlying reasons of the level of students' satisfaction perceived by the instructor identified in the previous question.
7. In terms of a course tool, what do you believe are the necessary features an "ideal" toolkit should possess?	Free-form text	To understand the most important features of agent toolkits from the instructor's point of view.
8. Do you plan to continue using / demonstrating agent toolkits in your course?	Yes / No	To help project a trend on the future use of agent toolkits in academic courses.
9. Why / why not?	Free-form text	Addresses why certain instructors will or will not use agent toolkits in their courses in the future.

Table 3: Questions asked of instructors who utilized agent toolkits

4.1.2. Participant Recruitment

Participants were recruited in two ways. The first was to conduct an intensive Web search for college and university instructors who taught agent-related courses. This involved visiting a number of discussion forums and web portals pertaining to agents, such as UMBC Agents[20] and MIT's Media Lab[21] sites. It also involved the use of search engines, such as Google and Excite, and Web directories, such as Yahoo! and Open Directory. In total, 256 potential participants were identified from over 500 agent-related web sites visited. Names and contact information of instructors were derived from the examination of roughly 300 online course syllabi. These instructors were contacted through a personalized email message that explained the purpose of the project and asked them to take a few minutes to complete the online questionnaire. Figure 5 below illustrates the Web site used to host the online questionnaire. A second follow-up e-mail was issued one month later to those instructors who did not fill out the questionnaire the first time. In total, 77 participants were recruited in this manner.

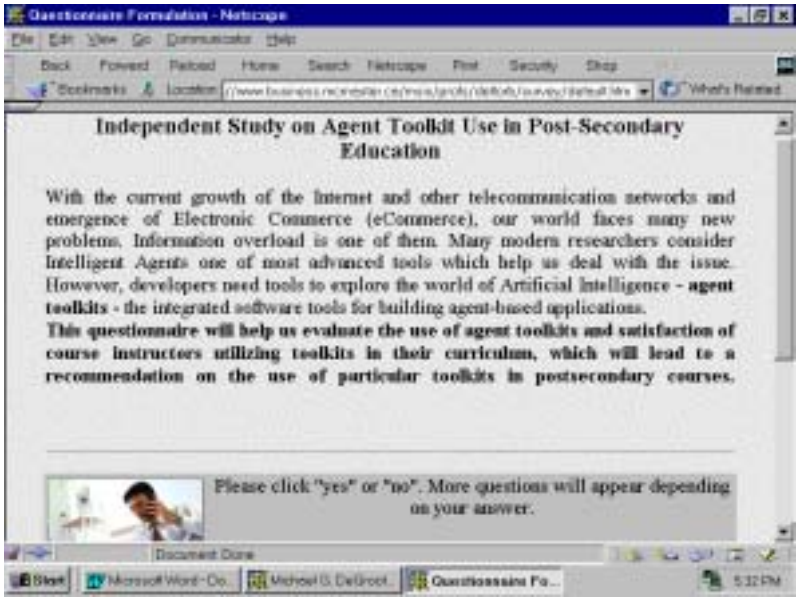

Figure 5: The first page of the Web-based questionnaire completed by participants

[20] See http://agents.umbc.edu

[21] See http://agents.media.mit.edu/index.html

The second was to solicit a general call for participants through the ISWorld[22] listserv. ISWorld is a Web portal site dedicated to servicing the needs of information systems academics across the globe. It was thought that course instructors having an interest in agent toolkit technologies would be members of this community. The authors had seen several call for papers for agent-related academic conference posted through the ISWorld listserv in the past. The call for participants was done on January 15th, 2002, reaching 2,927 recipients and yielding an additional 10 responses.

With these two methods, 87 questionnaires in total were completed. Though not an overly large number, the authors believe that a sizeable and random representation of academics with interests in utilizing agent toolkits in their curriculums was achieved. This statement is based on the authors' extensive Web search which found a limited number of course syllabi pertaining to agent technologies in general.

To analyze the collected data, descriptive statistics and t-tests were conducted on the quantitative portions of the questionnaire. Content analysis was performed on segments of the questionnaire where respondents replied with free-form text.

4.2. Results

Of the 87 respondents who filled out questionnaires, 25 utilized agent toolkits and 62 did not. The questionnaire results for each of these two sets of respondents are discussed separately in the following two sub-sections.

4.2.1. Instructors Who Utilized Agent Toolkits

Tables 4, 5, and 6 list the agent toolkits utilized by instructors by agent toolkit manufacturer. These manufacturers are commercial, academic, and non-profit institutions respectively. These are the same categories outlined previously in Section 3.5.

[22] See http://www.isworld.org

Agent Toolkit	Description	Developer	Contact Info
ABLE	Java framework, component library, and productivity tool kit for building intelligent agents using machine learning and reasoning	IBM T.J. Watson Research Center	www.research.ibm.com/able/index.html
AgentBuilder	Integrated Agent and Agency Development Environment	IntelliOne Technologies	www.agentbuilder.com
Grasshopper	Agent development platform	IKV ++ Technologies AG	www.grasshopper.de
IBM Aglets	Mobile agents development environment	IBM Japan	www.iks.com
JACK	A third-generation agent system that provides the architecture and capability for developing and running software agents in distributed applications	Agent Oriented Software	www.agent-software.com.au
JESS	Rule engine and scripting environment allowing building Java applets and applications that have the capacity to "reason" using knowledge you supply in the form of declarative rules	Sandia National Laboratories	herzberg.ca.sandia.gov/jess
MADKit	Java multi-agent platform built upon an organizational model	Madkit Development Group	www.madkit.org
Pathwalker	Distributed process-oriented and agent-oriented programming library	Fujitsu Laboratories Ltd	www.labs.fujitsu.com/free/paw
Zeus	Agent component library with visual development and visualization tools	British Telecommunications Labs	193.113.209.147/projects/agents.htm

Table 4: Commercial manufacturers of agent toolkits used by instructors

Agent Toolkit	Description	Developer	Contact Info
Agora	Provides a multi-agent architecture designed for supporting collaborative work in a distributed environment.	Norwegian University of Science and Technology	www.idi.ntnu.no/~agent/exercises/ex1/ex1_agora_intro.html
AIMA	Includes psuedo-code versions of all the major AI algorithms	Stuart Russell and Peter Norvig	Russell and Norvig Book www.cs.berkeley.edu/~russell/aima.html
DECAF	Offers a platform to design, develop, and execute agents.	University of Delaware	www.eecis.udel.edu/~decaf
JADE	Offers an environment for MAS implementation via middleware that complies with FIPA specifications. Offers a set of tools for agent debugging and deployment.	University of Parma	sharon.cselt.it/projects/jade
MAgnUM JAMAICA	Multi-agent intelligent systems.	Universidade do Minho	www.uminho.pt
MACE3J	A MAS simulation, integration, and development test bed that offers a supporting library of components, examples, and documentation.	Graduate School of Library and Information Science, University of Illinois	www.isrl.uiuc.edu/amag/mace/
MICE (The Michigan Intelligent Coordination Experiment)	Tool for experimenting with coordination between intelligent systems under a variety of conditions within a 2D grid world where agents operate and affect their environment.	University of Michigan	www.umich.edu/
RePast	Offers a software framework for creating agent based simulations.	University of Chicago's SSRC	repast.sourceforge.net
SimAgent	Provides research and teaching resources related to the development of interacting agents in environments of various degrees and complexities.	The University of Birmingham School of Computer Science	www.cs.bham.ac.uk/~axs/cog_affect/sim_agent.html
Soar	Supports a cognitive architecture for systems with intelligent behavior.	University of Michigan	ai.eecs.umich.edu/soar
TeamBots	Tool for building autonomous robots for cooperative tasks.	Carnegie Mellon University	www.teambots.org
Wumpus World Simulator	Allows users to write arbitrarily complex agents that reason and exist within an agent environment.	University of Texas at Arlington	www-cse.uta.edu/~holder/courses/cse5361/wumpus.html

Table 5: Academic manufacturers of agent toolkits used by instructors

Agent Toolkit	Description	Developer	Contact Info
FIPA-OS	Component-based toolkit enabling rapid development of FIPA compliant agents	Emorphia Ltd	fipa-os.sourceforge.net
GA Playground	General purpose genetic algorithm toolkit where the user can define and run his own optimization problems		www.aridolan.com/ga/gaa/gaa.html

Table 6: Non-profit manufacturers of agent toolkits used by instructors

Tables 4, 5, and 6 identify 23 different toolkits used by the 25 respondents who utilized agent toolkits in their courses. Most instructors stated they used only one toolkit in their courses; a few identified a couple of toolkits. JADE was the most popular toolkit utilized in that it was identified by 5 instructors. Overall, these findings suggest that a wide variety of toolkits were utilized by instructors, and that, with the exception of JADE, there was no definitive toolkit utilized by a sizeable percentage of instructors. It appears that instructors may acquire toolkits that best match an instructor's specific course requirements and that no single "uniform toolkit" exists that may meet the specific needs of all agent-related courses.

Figure 6 below summarizes the results of Tables 4, 5, and 6. The figure shows the percentage breakdown between academic, commercial, and non-profit manufacturers. More than half the instructors who utilized agent toolkits in their courses used toolkits developed by academic manufacturers.

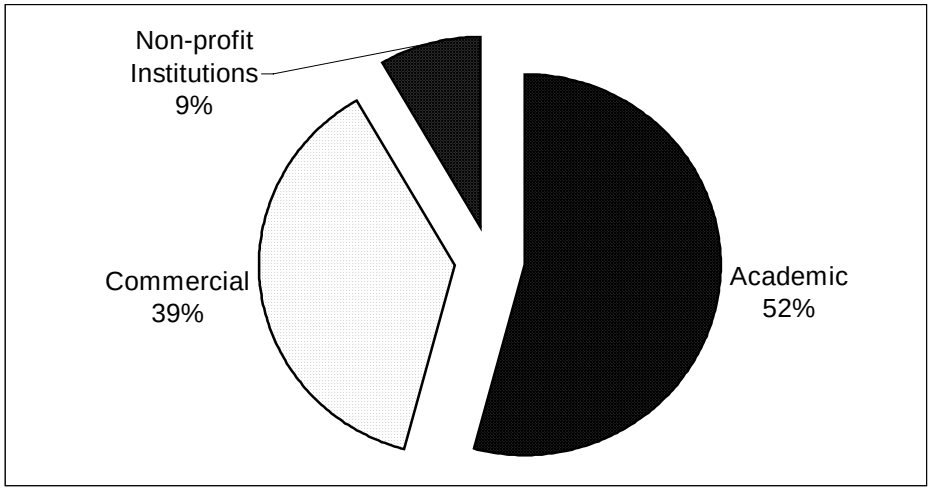

Figure 6: Manufacturers of agent toolkits used by instructors

Table 7 below lists the agent toolkits according to the name of the agent course in which they used. The categories of agent course names were devised from a content analysis of agent course titles identified by instructors on the questionnaire. Figure 7 summarizes the results of Table 7 by showing the percentage breakdown of agent toolkit usage by course name category. Note some toolkits appear across more than one course name categories (i.e., they were not utilized in just one type of course).

Course Name	**Agent Toolkits Utilized**
Intelligent agents	AgentBuilder, Agora, IBM Aglets, Grasshopper, JADE, Pathwalker
Multi-agent and multi-robot systems	ABLE, DECAF, FIPA-OS, Jade, Jack, MACE3J, MADKit, Pathwalker, RePast, TeamBots
Agent technologies	JADE, JESS, ZEUS
Artificial intelligence (Distributed AI)	Agora, JACK, MICE, SimAgent, Wumpus World Simulator
Knowledge-based systems	Soar
Machine learning	GA Playground
Information gathering	DECAF

Table 7: Types of courses in which agent toolkits were used

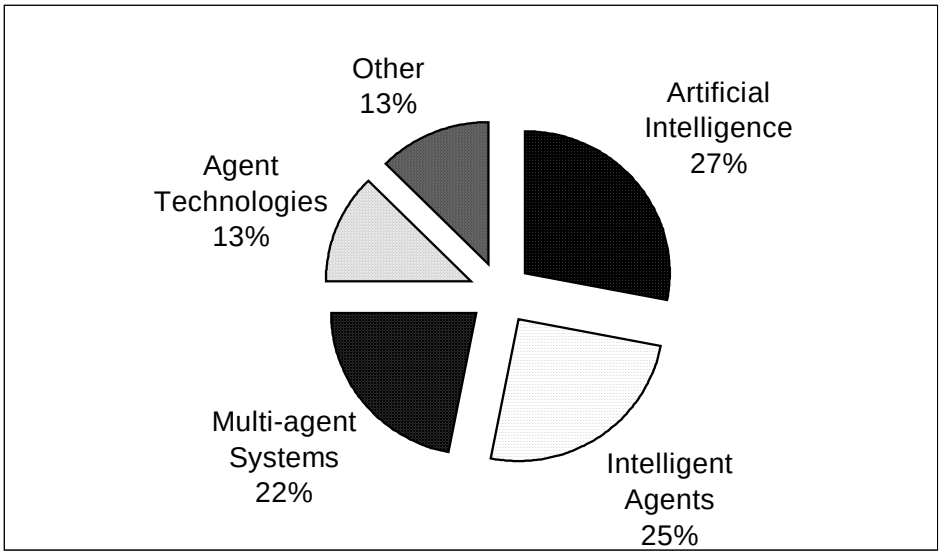

Figure 7: Types of courses in which agent toolkits were used

A major distinction between courses was the degree to which the course focused on artificial intelligence (AI). Dividing the courses in this way, it was found that AI focused courses constituted 36% of all courses in which agent toolkits were utilized, and that non-AI focused courses comprised 64%.

Recall that one of the objectives of this project was to identify the satisfaction of instructors who utilized agent toolkits in their courses. Two satisfaction questions were asked: one concerning instructors' own level of satisfaction; and another about their perception of student satisfaction with the toolkits. Scores for these questions utilized a Likert scale of 1 to 4 (1 being "not satisfied" and 4 being "very satisfied"). Though there were some deviations in the levels of satisfaction rated by instructors for these two questions, in general the scores given for these two questions were usually identical. The response scores from these two questions were averaged into a single index of toolkit satisfaction. Across all instructors, the average satisfaction index score was 2.59 (somewhat satisfied / satisfied).

In terms of AI course orientation, it was found that the average satisfaction level of AI focused courses was 2.98 and that of non-AI courses was 2.47. A t-test confirmed the average satisfaction level of AI focused courses was higher than non-AI focused courses ($p<.05$).

In terms of toolkit manufacturer, it was found that the average satisfaction level of academically-developed toolkits was 2.83 and with commercially-developed toolkits was 2.36. A t-test confirmed the average satisfaction level of academically-developed toolkits was higher than that of commercially developed ones ($p<.05$).

According to Human Computer Interaction (HCI) research, three viable measures of software usability are functionality, performance, and user interaction. Functionality refers to *what* functions and features a piece of software provides. Performance relates to *how well* the software delivers these functions (e.g., response times; storage capacity). User interaction refer to the *ease* with which users can instruct the software to perform its tasks and the degree to which users can readily engage themselves with the software. It is assumed that the more usable a piece of software is, the more satisfied users would be. With regards to agent toolkits, which are a specific type of software, it follows that the more usable an agent toolkit is in terms of its

functionality, performance, and interaction, the more satisfied instructors would be with utilizing the toolkit in their courses.

Utilizing these three measures of usability as a guide, a content analysis was performed on the free-form text responses made by participants in the questionnaire for the two items which asked instructors to explicate the reasons behind their indicated levels of satisfaction with the toolkits.

Figure 8 displays the overall breakdown of this content analysis for those comments elicited by participants which implied *satisfaction* with the toolkit. Roughly half of the responses pertained to user interaction characteristics (e.g., ease of use; comprehensiveness of the software's documentation; short learning curve; toolkit familiarity; and the user-friendliness of the software interface). Almost one third of the responses related to performance characteristics (e.g., performed well; stability of the software; powerfulness of the toolkit). Over one fifth of the responses pertained to functionality (e.g., the software supported what users wanted).

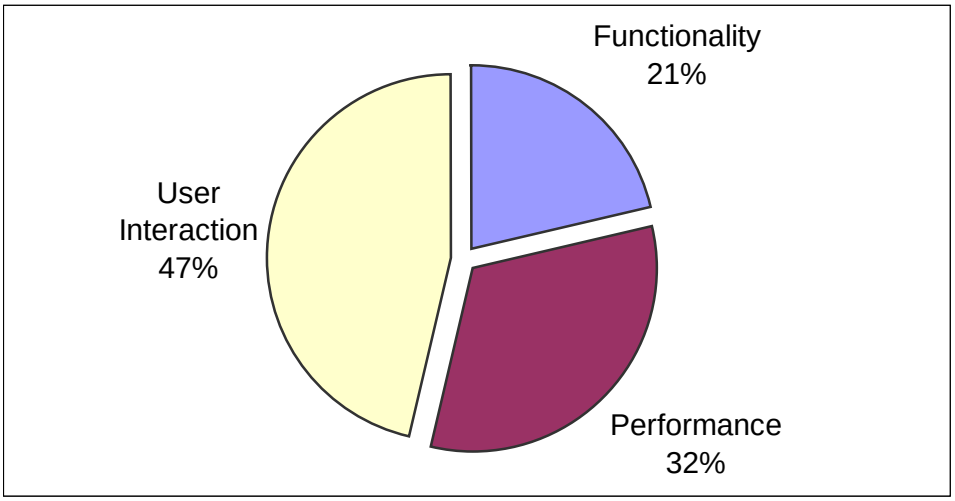

Figure 8: High-level reasons why instructors were satisfied with the toolkits

Table 8 below shows the more detailed reasons why instructors were satisfied with the agent toolkits they utilized across the three measures of functionality, performance, and user interaction.

Usability Factor	Detailed Reason	Percentage of instructors who mentioned this reason (N=25)
Functionality	Employment of all necessary functions and features (e.g. employment of basic agent concepts)	24%
Performance	Performed well	16%
	Stability	12%
	Powerfulness	8%
User Interaction	Ease of use	16%
	Comprehensiveness of the toolkit's documentation	12%
	Short learning curve	12%
	Familiarity with the workings of the toolkit (e.g. software was developed locally; software had been used in the past)	8%
	User-friendliness of the GUI	4%

Table 8: Reasons why instructors were satisfied with the agent toolkits

Figure 9 displays the overall breakdown of the comments elicited by participants which implied *dissatisfaction* with the toolkit. Again, almost half of the responses pertained to user interaction characteristics (e.g., high complexity; substantial degree of programming required; poor software documentation). However, the ratios for functionality and performance were reversed. Almost one third of the responses related to functionality characteristics (e.g., lack of important functions and features). Over one fifth of the responses pertained to performance (e.g., not properly working features; instability).

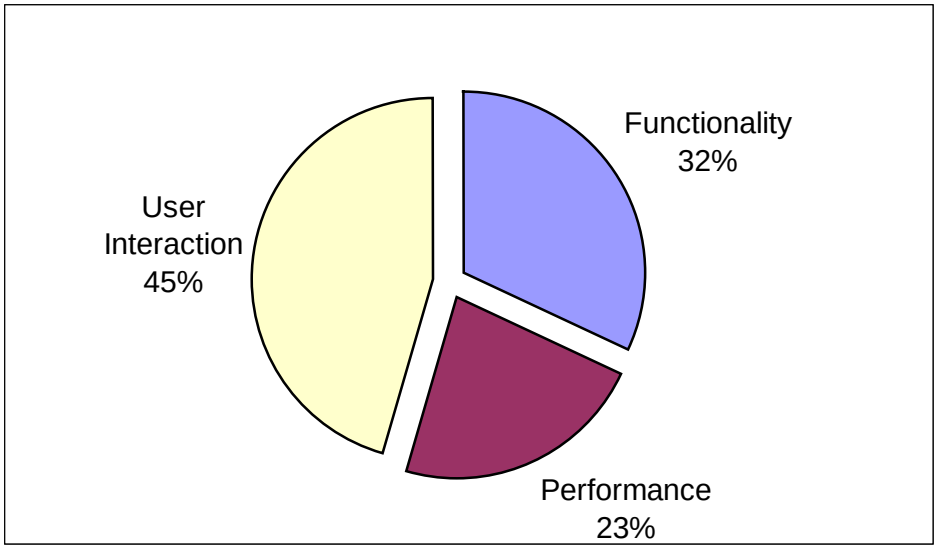

Figure 9: High-level reasons why instructors were dissatisfied with agent toolkits

Table 9 below shows the more detailed reasons why instructors were dissatisfied with the agent toolkits they utilized across the three measures of functionality, performance, and user interaction.

Usability Factor	**Detailed Reason**	**Percentage of instructors who mentioned this reason (N=25)**
Functionality	Lack of important functions and features (e.g. Forward and Back reasoning, no support for agent interactions. Some instructors and students had to implement these functions by themselves)	28%
Performance	Not properly working features	12%
	Instability	8%
User Interaction	High complexity	20%
	Substantial programming efforts	12%
	Poor documentation	8%

Table 9: Reasons why instructors were dissatisfied with agent toolkits

There were several recurring comments made by instructors in terms of the features an "ideal" agent toolkit should possess. First, a toolkit should provide a persistent, flexible, robust and reliable environment in which agents can be programmed. The environment should respect and

accommodate different levels of student computer programming abilities. Further, the environment should support artificial intelligent capabilities for agents, such as knowledge representation and problem solving.

Second, the toolkit should be user-friendly. Clear graphical user interface, debugging and simulated environments would help visualize activities and help students better understand agent concepts.

Third, a number of teaching aids should be provided. This includes features such as: examples of working, collaborating agents along with their sample code; agent building templates; and strong supporting user documentation. Such tools are critical for student success with agent toolkits.

Last, the toolkit should be relatively inexpensive and easy to install to facilitate wide adoption and use throughout academia.

Table 10 summarizes these factors in terms of the usability factors identified above. Figure 10 shows the high-level breakdown.

Usability Factor	Detailed Reason	Percentage of instructors who mentioned this reason (N=25)
Functionality	Offers all necessary functions for a course	16%
	Provides an environment supportive of agent creation	4%
Performance	Is stable	20%
	Offers flexibility	8%
	Is easy to install	4%
User Interaction	Provides sample agents (with code) and demos	32%
	Offers strong supporting documentation	28%
	Is easy to use	28%
	Provides a user-friendly GUI	28%
	Supports various levels of student programming knowledge	20%
	Provides debugging and simulated environments	12%
	Offers many adaptable levels of difficulties	8%
Additional	Is inexpensive	4%

Table 10: Desirable features of an ideal toolkit expressed by instructors

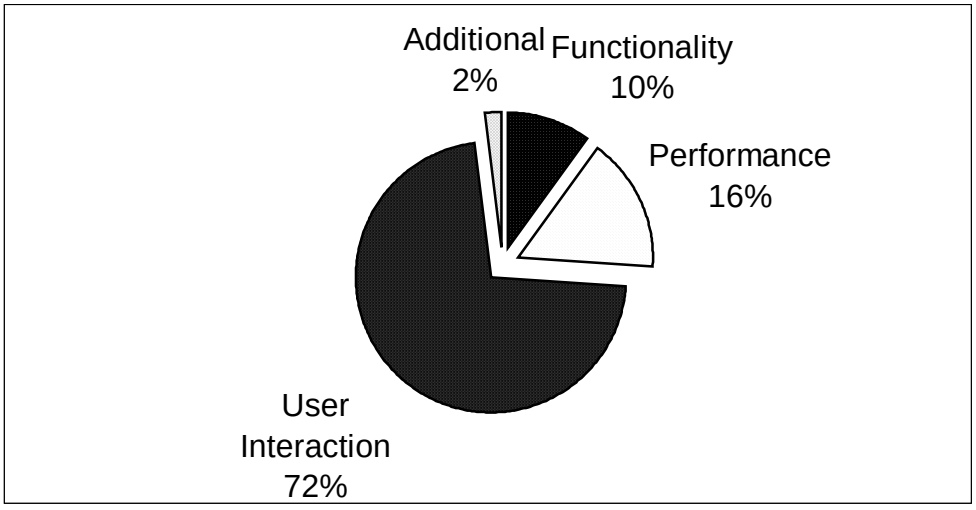

Figure 10: Usability factors deemed desirable by instructors in an agent toolkit

The findings from Figures 8, 9, and 10 are summarized in Table 11 below. The table demonstrates the relative importance of user interaction characteristics in agent toolkits, as expressed by instructors utilizing toolkits in their courses.

Usability Factor	**Satisfaction Reasons**	**Dissatisfaction Reasons**	**"Ideal" Toolkit Characteristics**
Functionality	21%	32%	10%
Performance	32%	23%	16%
User Interaction	47%	45%	72%

Table 11: Summary of instructor responses to usability characteristics in agent toolkits

Of the 25 instructors who utilized agent toolkits in their courses, 88% stated they would utilize such toolkits again. These instructors believed that the toolkits enriched their programs, forming an integral part of the course and providing a useful aid for teaching students about agent behavior. The instructors stated that the toolkits fostered student understanding of agent programming, technologies, and concepts by allowing students to implement their own agents. Such tasks encouraged students to think at a higher levels of abstraction. It was also believed that students gained valuable hands-on experience with toolkits they likely would confront when working in industry. The 12% of instructors who stated they were not going to continue utilizing agent toolkits in subsequent courses, were either "not satisfied" or "somewhat satisfied" with their toolkit experience that resulted from either poor performance, limited functionality or toolkit instability.

4.2.2. Instructors Who Did Not Utilize Agent Toolkits

In contrast to the group of instructors discussed in the previous section, this set of questionnaire respondents were instructors of post-secondary agent-related courses who did not utilize agent toolkits. Figure 11 illustrates the breakdown of responses made by this group of instructors on whether agent toolkits would be beneficial to use.

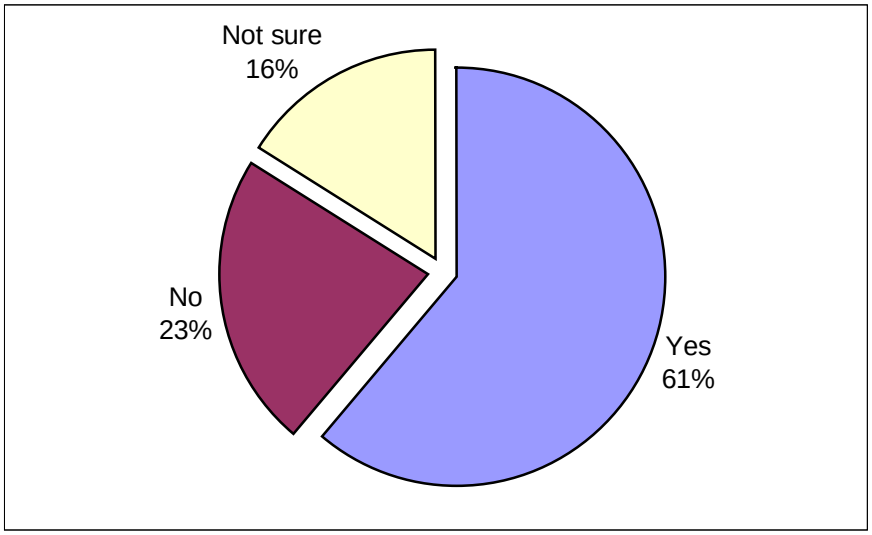

Figure 11: Breakdown of instructors who thought agent toolkits would be beneficial to use in the classroom

As the diagram shows, the majority (61%) believed that agent toolkits would be beneficial to use in the agent-related courses they taught. There are several reasons why. First, given the importance of agents and agent-related research in today's Internet world, these instructors felt it was critical that students not only understand the fundamentals and underlying theory pertaining to agents, but also modern agent technologies and tools.

Second, these instructors believed that theoretical examples were insufficient for students and that they needed working examples of agents to convince them of the usefulness of agent technologies and research. They also believed that hands-on experience with the design, creation, and implementation of agents was the best way of enabling student comprehension of agents and of fostering student interest in agent-related research.

Third, these instructors felt that agent toolkits would allow students to concentrate more on agent issues (such as the actions an agent must perform based on another agent's communications)

rather than lower-level technical issues (such as writing code to parse packets of data sent in agent communication). Lower level technical issues tend to involve too much programming and distract students from understanding higher-level agent concerns taught in the course material.

Despite this enthusiasm for toolkits, these instructors cautioned that toolkits be used with care. They mentioned that toolkits were like any software teaching aid and need to be introduced and utilized in courses appropriately. That is, they should be used in the right context with the right material and in the right circumstances.

Of the 23% of instructors who thought that toolkits would offer little benefit to students, the majority framed this comment in context of the current agent-related course they were teaching. These tended to be introductory agent courses. Here, the instructors felt the time students would spend learning the toolkit would be better utilized on other parts of the course which placed more emphasis on basic concepts and fundamentals. A few instructors commented that agent toolkits would be better utilized in more advanced agent courses.

The remaining 16% of instructors who were unsure whether agent toolkits would be beneficial or not in the classroom stated that more research was required to investigate this issue further.

When asked why toolkits had not been utilized so far, a variety of reasons were given. Figure 12 below summarizes these answers.

The predominant response was that toolkits were inappropriate for the course being taught. This pertained primarily to basic or introductory courses where most time is spent on teaching fundamentals rather than agent engineering, or where the length of the course is short (e.g., six weeks in duration) limiting the amount of time instructors can dedicate to implementing agent technologies in the classroom. Most respondents considered creating agents an advanced proposal in their courses.

The second most popular response was that the instructors were unaware of the existence of agent toolkits.

This was followed by the third most popular response by instructors, who were very familiar with the existence of agent toolkits but failed to find an appropriate one that matched their own

course requirements. Many lamented the extensive reliance on advanced programming techniques in the current batch of available toolkits on the market, making the toolkits inappropriate as teaching aids for students without extensive programming knowledge. Despite these instructors' inability to find an appropriate toolkit after conducting an extensive search for one, all believed that the usage of toolkits would be beneficial for their courses.

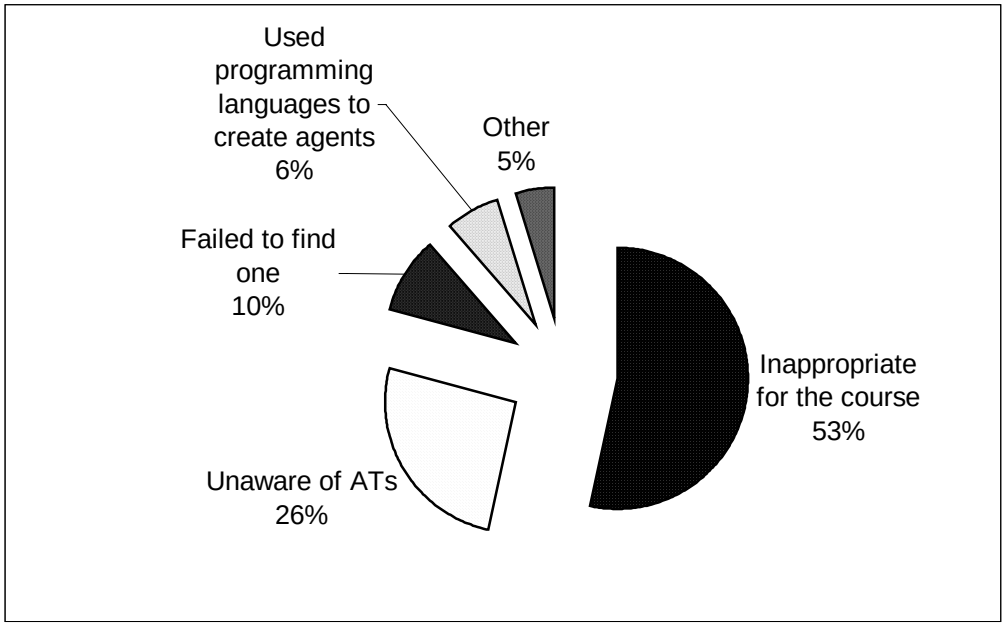

Figure 12: Reasons why instructors did not use agent toolkits

The fourth response was by a group of instructors who allowed students to create intelligent agents in a programming language of their own choice. These instructors did not want to restrict their students to any particular language or application. In this case, students had to program features in their agents that toolkits would have provided, such as the ability to communicate and send messages to other agents.

The remaining responses covered a wide range of concerns. Here, instructors had not yet investigated the use of agent toolkits for their courses or felt they did not have enough knowledge to include one.

5. Discussion and Conclusions

Recall the purpose of this project was twofold. The first was to conduct a general comparison of agent toolkits on the market in terms of the functionality they offer and their underlying technology. The second was to evaluate the use and satisfaction of course instructors utilizing agent toolkits in their curriculums.

In terms of agent toolkits on the market, the study's findings were based on an extensive search and downloading of agent toolkits. A comparison of the downloaded toolkits yielded four major categories of agent toolkits: mobile agent toolkits; multi-agent toolkits; general purpose toolkits; and Internet agent toolkits. The authors caution that this categorization schema is not finalized and most likely will need to be adjusted in the future given the dynamic nature of agent development environments. Of the toolkits examined, more were developed by academic institutions. In terms of underlying technology, most utilized Java, due to its platform independence, followed by Python, Tcl, C/C++ and Prolog. All toolkits that were analyzed differed considerably from each other in terms of functionality depending on their area of application. Some toolkits required users to possess advanced programming skills, which must be taken into consideration when choosing a toolkit for consideration as an educational tool.

With respect to instructor use and satisfaction with agent toolkits in the classroom, the study showcased some interesting findings. These were based on responses from 87 participants who filled out an online questionnaire. The participants were randomly chosen and formed a representative sample population of instructors teaching agent-related courses.

The research demonstrated that a minority of instructors teaching agent-related courses currently included agent toolkits in their curriculums. Of toolkits that were used, no single uniform toolkit met the specific needs of all agent-related courses. Rather, a wide variety of toolkits were utilized across disparate types of agent-related courses. More than half of the toolkits used were developed by academic manufacturers. "Intelligent agents" and "multi-agent systems" courses formed the predominant category of courses employing agent toolkits. Two-thirds of the courses that utilized toolkits were non-AI focused.

Overall, the average rating of toolkit satisfaction ranged between “somewhat satisfied” to “satisfied”. The average satisfaction level of instructors was higher for AI focused courses and for toolkits developed by academic manufacturers. The level of satisfaction and decision whether to continue using a toolkit in a course depended upon an instructor’s personal experience utilizing that toolkit in the classroom. The factors influencing instructor satisfaction were: 1) the toolkit’s functionality – whether the toolkit matched the instructor’s needs, 2) the toolkit’s performance – whether the toolkit was capable of implementing all advertised functions; and 3) the toolkit’s usability – whether it offered a user-friendly, positive experience.

User interaction was the leading characteristic of agent toolkits that influenced satisfaction levels over that of toolkit performance and functionality. In terms of user interaction, satisfaction levels were higher when the toolkit was easy-to-use (i.e., simple, non-programming intensive, quick to learn, accompanied by a comprehensive documentation set). With respect to performance, satisfaction levels were higher when the toolkit operated well (i.e., was stable, was powerful). In terms of functionality, satisfaction levels were higher when the toolkit supported basic agent concepts.

Instructors identified characteristics of an ideal toolkit. They suggested the toolkit should provide a persistent, reliable, flexible, easy-to-use environment for agent creation and deployment. They also stated the toolkit be accompanied by several teaching aids, such as working examples, sample code, and agent generation templates, as well as being affordable and easy-to-install. The majority of instructors who did not currently utilize toolkits in the classroom, felt that toolkits would be beneficial for their students.

It should be noted that agent toolkits are in the early stages of development. The pace of adoption, however, is constantly increasing. We believe that more instructors will adopt toolkits in the classroom as agent toolkits become more easy-to-use, perform better, and support a wider range of agent-related functions, and as instructors themselves become more knowledgeable about agent toolkits on the market. We also believe that no particular toolkit will be appropriate for all instructors – the chosen toolkit needs to match individual course requirements and instructor preferences.

We are encouraged by the future use of agent toolkits in post-secondary education. Agent toolkits are useful and practical applications for creating, deploying, and re-using agents. Though the toolkits are not at a stage of development yet which yields enthusiastic satisfaction scores by instructors of agent-related courses, the toolkits currently available on the market do offer distinct advantages over other software development environments. Most current development platforms, packages and compilers do not allow creating software components with agent capabilities, such as personalization, productiveness, adaptiveness and proactiveness. They also do not address implementation of other required agent features, such as agent interaction rules, agent communication, and common knowledge bases. Agent toolkits address these concerns by providing a conceptual level of abstraction, supporting agent functionality, and offering run-time testing and debugging environments.

There were several limitations of the study that should be mentioned. First, the small sample size of instructors who actually utilized agent toolkits in the classroom makes the study's finding difficult to generalize to the larger population of instructors of agent-related courses. There were also difficulties in installing some of the toolkits which prohibit a more thorough analysis of toolkits on the market. There were also challenges in comparing the usability of the toolkits across the different toolkits downloaded as each toolkit was unique and designed for distinct, special purposes.

However, despite this limitations, the study yielded pertinent preliminary findings which provide a good test bed for future research in this area. Several avenues can be explored. For example, the level of satisfaction of students can be investigated by surveying the students directly (and not just the instructors). It would be interesting to see if student satisfaction levels differed significantly from instructors, and more importantly, what usability factors affect student satisfaction levels. Another idea would be to poll industry participants utilizing agent toolkits and determine those criterion which affect satisfaction. Satisfaction levels for particular toolkits could be explored, perhaps by polling instructors who use the same toolkit in similar courses. The available agent toolkits on the market should be further investigated. Over 40 agent toolkits were discovered in this research. However, this list may be extended and new categories of toolkits can be defined. Doing so would facilitated better comparison of agent toolkits in terms of their functionality, underlying technology, and ease of use.

Overall, this report has shed light on the current landscape of agent toolkits on the market and their use in post-secondary education. It is the hope of the authors that more research and exploratory investigations will be done in this area.

6. References

AgentBuilder (2000). *The agentbuilder user guide*. San Diego, California: Reticular Systems.

Bellifemine, F., Poggi, A., Rimassa, G., Turci, P. An object-oriented framework to realize agent systems, Proceedings of WOA 2000 Workshop, Parma, May 2000, pp. 52-57.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 34-43.

DeLoach, Scott (2001). Analysis and design using MaSE and agentTool. *Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference*, Miami University, Oxford, Ohio.

Gray, Robert S., Cybenko, George, Kotz, David, & Rus, Daniela (2000). *Mobile agents: Motivations and state of the art.* (Technical Report TR2000-365). Department of Comupter Science, Dartmouth College (Hanover, New Hampshire, USA).

Jennings, N. R., & Wooldridge, M. J. (1998). Applications of intelligent agents. In N. R. Jennings & M. J. Wooldridge (Eds.), Agent Technology Foundations, Applications, and Markets. Berlin: Springer.

Lange, Danny (2002). *Java aglets application programming interface*, (White Paper). Available at http://www.trl.ibm.com/aglets/JAAPI-whitepaper.html (last accessed June 2002).

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM* 37(7).

Maes, P. (2001). Smart commerce: The future of intelligent agents in cyberspace. In P. Richardson (Ed.), *Internet Marketing* (pp. 139–46). New York: McGraw-Hill.

Maes, P., Guttman, R., & Moukas, A. (1999). Agents that buy and sell: Transforming commerce as we know it. *Communications of the ACM* 42(3), 81–91.

Sinmao, M.V. (1999). *Intelligent agents and e-commerce*. The McDonough School of Business - Georgetown University, MGMT 681 Course - Electronic Commerce Research Briefings. Available at www.msb.georgetown.edu/faculty/culnanm/EC/Briefings2/Sinmaom.htm (last accessed June 2002).

Port, O. (2002). The next web. *Business Week Online, March 4*. Available at http://www.businessweek.com/magazine/content/02_09/b3772108.htm (last accessed July 2002).

Rahman, S. M., & Bignall, R. J. (Eds.). (2001). *Internet commerce and software agents*. Hershey, Pennsylvania: Idea Group.

Russell, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Upper Saddle River, New Jersey: Prentice-Hall.

Shoham, Y. (1997). An overview of agent-oriented programming. In J. M. Bradshaw (Ed.), *Software Agents* (pp. 271–90). Menlo Park, Calif.: AAAI Press.

Turban, E., Lee, J., King, D., & Chung, H.M. (2000). *Electronic commerce: A managerial perspective*. Upper Saddle River, New Jersey: Prentice Hall.

Michael G. DeGroote School of Business, McMaster University

WORKING PAPERS – RECENT RELEASES

427. Halit Üster and Robert F. Love, "Application of Weighted Sums of Order *p* to Distance Estimation", April, 1998.

428. Halit Üster and Robert F. Love, "On the Directional Bias of the l_{bp}-norm", April, 1998.

429. Milena Head, Norm Archer, and Yufei Yuan, "MEMOS: A World Wide Web Navigation Aid", October, 1998.

430. Harish C. Jain and Parbudyal Singh, "The Effects of the Use of Strike Replacement Workers on Strike Duration in Canada", February, 1999.

431. Parbudyal Singh and Harish C. Jain, "Strike Replacements in the United States, Canada and Mexico: A Review of the Law and Empirical Research", February, 1999.

432. John W. Medcof and Jeremy Boyko, "Reinforcing, Revising and Reconciling Attributions in the Employment Interview", March, 1999.

433. Norm Archer, "World Wide Web Business Catalogs in Business-to-Business Procurement", March, 1999.

434. Diwakar Gupta and Saifallah Benjaafar, "Make-to-order, Make-to-stock, or Delay Product Differentiation? - A Common Framework for Modeling and Analysis", April, 1999.

435. Harish C. Jain, Parbudyal Singh and Carol Agocs, "Recruitment, Selection and Promotion of Visible Minorities and Aboriginals in Selected Canadian Police Services", April, 1999.

436. Harish C. Jain and Angus Bowmaker-Falconer, "Employment Equity/Affirmative Action Codes of Practice and Best Practices in USA, Britain, Canada and Other Selected Countries", May, 1999.

437. Diwakar Gupta, Yavuz Günalay, and Mandyam M. Srinivasan, "On the Relationship Between Preventive Maintenance and Manufacturing System Performance", June, 1999.

438. Jinliang Cheng, George Steiner, and Paul Stephenson, "A Fast Algorithm to Minimize Makespan for the Two-Machine Flow-Shop Problem with Release Times", June, 1999.

439. Jinliang Cheng, George Steiner, and Paul Stephenson, "A Fast Algorithm to Minimize Maximum Lateness for the Two-Machine Flow-Shop Problem with Release Times, June, 1999.

440. Norm Archer and Yufei Yuan, “Electronic Commerce and the Business-to-Business Customer Relationship Life Cycle”, August, 1999.

441. Halit Üster and Robert F. Love, “Calculation of Confidence Intervals for Estimated Distances”, August, 1999.

442. George O. Wesolowsky, “Detecting Excessive Similarity in Answers on Multiple Choice Exams”, October, 1999.

443. Naresh C. Agarwal, “Mandatory Retirement and the Canadian Human Rights Act”, November, 1999.

444. Susan Sproule and Norm Archer, “Software Agents in Electronic Commerce”, March, 2000.

445. Milena Head and Yufei Yuan, “Privacy Protection in Electronic Commerce – a Theoretical Framework”, June, 2000.

446. Susan Sproule and Norm Archer, “Knowledgeable Agents for Search and Choice Support in E-commerce: A Decision Support Systems Approach”, July, 2000.

447. Norm Archer and Judith Gebauer, “Managing in the Context of the New Electronic Marketplace”, August, 2000.

448. Norm Archer and Yufei Yuan, “Business-to-Business E-commerce and Customer Relationship Management: Trends and Issues”, August, 2000.

449. Janet Romaine and I.U. Zeytinoglu, “Are Women Managers Really More Participative? Some Conflicting Evidence from a Field Study”, September, 2000.

450. Norm Archer, “The New Economy: Some Issues and Impacts of Electronic Commerce”, January 2001.

451. Norm Archer, Milena Head, and Yufei Yuan, “Matching Customer Expectations and Perceptions in e-Commerce”, February 2001.

452. Xiaoqing Li and Ali R. Montazemi, “A Methodology for the Assessment of Buddy-Agents”, March 2001.

453. Yufei Yuan, T.P. Liang, and Jason J. Zhang, “Using Agent Technology to Support Supply Chain Management: Potentials and Challenges”, October 2001.

454. Jason J. Zhang, Yufei Yuan, and Norm Archer, “Driving Forces for M-commerce Success”, December 2001.