Using Mobile Agents in Real World: A Survey and Evaluation of Agent Platforms

Josef Altmann, Franz Gruber, Ludwig Klug, Wolfgang Stockner, Edgar Weippl
Software Competence Center Hagenberg
Hauptstr. 99
A-4232 Hagenberg, Austria
+43 7236 3343 800
{firstname.lastname}@scch.at

ABSTRACT

In the last few years agent technology, especially mobile agents, became a new exciting field in computer science. Agent development is getting more and more interesting, even in commercial infrastructures, so it is worth considering in depth their strengths and the situations in which they can be used effectively. During our research work we figured out, that there exist a huge number of approaches, toolkits, and platforms of different quality and maturity. This fact led us to the problem of evaluating them to find an "optimal" (in the sense of productivity) platform.

Firstly, to get an impression what should be possible for an agent platform, we collected the current research results and summarized them in a state-of-the-art report. After that, we defined a set of criteria, which collected the requirements to the platforms. Then we selected some fundamental criteria in a way that if one platform does not fulfill these criteria it is immediately knocked out from further evaluation. For the remaining platforms we applied the complete set of criteria, weighted them, and got a final evaluation result.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: **Distributed Artificial Intelligence** – *Intelligent agents, Languages and structures*

General Terms

Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Standardization, Languages.

Keywords

mobile agents; intelligent agents; agent platform evaluation

1. INTRODUCTION

For the last couple of years mobile agents have been a thrilling and challenging area of research. The surge of recent activity in mobile agents has resulted in a plethora of agent development toolkits and platforms [1]. Many of them are research projects with huge differences in support, stability, and documentation. Even among the commercial platforms there are some that are not likely to support future versions of Java.

As the variety of advantages offered by agents are commonly known ([9] is a good starting point and provides an excellent overview; another interesting source could be [6], which deals with research efforts on agent technology funded by the European Union) one may be inclined to build large-scale projects on existing agent platforms. However, relying on possibly faulty and no longer maintained basic building blocks is a risk not to be underestimated. The obvious approach to mitigate the risk is to thoroughly assess various platforms. With this paper we try to provide you, with a checklist-like concept of how to efficiently and effectively perform such an assessment. We will repeatedly point to our findings based on a comparison made in Nov 2000. [5], [7] and [8] contain an extensive report, exceeding sixty pages, on a comprehensive comparison of Java agent platforms. However, as research on mobile agents continuously advances, our findings are most likely to be partially outdated by the time this article is published. Therefore we think the main value for the

reader lies in having the conceptual tools for updating and tailoring the assessment to his/her needs.

2. A QUICK OVERVIEW

In this section we will give an overview of how to plan and execute an assessment of agent platforms. All essential issues will be described in the following section, omitted details can be found in [8]. Based on this procedure we evaluated and ranked thirty agent platforms so that but a handful remained short-listed for thorough analysis.

The first step is to define criteria upon which the evaluation will be based. Once these criteria are defined, they have to be elaborated in greater detail by defining aspects for each criterion. Thereafter knockout criteria (and aspects) should be specified and weights assigned to all others to allow results to be ranked. Finally, the last step of the preparation is to compile a list of agent platforms to be evaluated.

As the evaluation process commences knockout criteria are applied to eliminate all platforms that obviously require no further attention. The remaining platforms are ranked and having found a cut-off threshold the top platforms are short-listed for in-depth analysis.

The remainder of the paper is structured as follows. Section 3 will discuss evaluation criteria we suggest to use, their various aspects, and how to measure them. Section 4 will give an overview over the evaluation process and highlight our findings that lead selecting one specific agent platform (e.g., the "winning" platform). However, we want to emphasize that the entire selection process was obviously based on the requirements we specified; other projects requiring an adaptation of various aspects and possibly also some criteria will find other agent platforms to be better suited. Moreover, our evaluation is a snapshot of platforms taken in November 2000.

3. EVALUATION CRITERIA

Before describing the evaluation criteria in detail we will briefly define fundamental terms used throughout this paper. We understand measurement method as the kind of technique used to evaluate criteria. Possible techniques include but are not limited to analysis of user licenses, existing documentation, Web pages describing products as well as techniques, and empiric tests.

A criterion can be broken up into aspects containing questions to be answered during evaluation. For instance, the criterion 'environment' may be divided into the aspects 'client', 'server', and 'network'. For each of these aspects, there are one or more questions to be answered. Questions used for evaluating aspects can be distinguished by the type of answer expected. A binary question has to be answered either by 'yes' or 'no'. Questions may be answered using marks ranging from '1' to '4', with '1' being the best value and '4' the worst. Obviously, the range can be arbitrarily chosen to fit an organizations current practice. Furthermore, some questions require a value as answer, such as for example the price of an agent platform.

In the following subsections we will elaborate criteria such as security, availability, environment, development, characteristic properties of agent platforms, and knockout criteria.

3.1 Security

Security encompasses a variety of security mechanisms including encryption, public key infrastructure (PKI), and support for executing signed code. The criterion describes the available mechanisms guaranteeing a secure execution environment of an agent development platform. Additionally, data encryption capabilities, and code signing is part of this criterion.

A detailed study of agent platforms and programming language documentation (e.g., the measurement method) should focus on five aspects.

Authentication (binary): Are there any mechanisms to provide authentication of agents? Can one trust code that has been migrated?

Encryption of data (mark): Can code and data be transported in encrypted form? Is it possible to identify whether code was modified while migrating?

Certificates (mark): Does the platform support certificates including deployment of certificates?

Authorization (mark): Does the platform include or support a role-based authorization system?

Access restriction (mark): Does the platform provide mechanisms for a flexible access control to agent and agent system resources (e.g., the file system)?

3.2 Availability

Availability, as we understand it, includes both preconditions and legal as well as financial aspects regarding deployment of a platform. As measurement method we propose to analyze user licenses and/or sales contracts, platform documentation, time required for installation, and effort required for maintenance. For our study we closely looked at four aspects.

Evaluation version (binary): Is there an evaluation version available?

Platform state (mark): What is the development state of the product (alpha, release, phase out, application examples)? This aspect is evaluated as follows: "1" - production release, "2" - beta release, "3" - early beta release, and "4" - phase out or pending (e.g., not supported or improved anymore).

3.3 Environment

The criterion describes what environmental preconditions must be met to successfully use an agent development platform in an

existing infrastructure. To measure the five aspects of this criterion we relied on documentation shipped with the platform and supporting Web pages.

Documentation (binary): Is the agent platform well documented? Consider the following questions: What does the documentation comprise (security, development, debugging, etc.)? For which target groups (users' guide, programmers' guide, etc.) is documentation available?

Supported operating systems (binary): Is the agent environment platform operating system independent?

3.4 Development

The criterion 'development' evaluates how efficient programs for the agent platform in question can be designed, implemented, and tested. Wizards, for example, would facilitate the process of creating new agents. To measure this mission-critical criterion, we will go beyond studying platform and programming documentation to include hands-on experience from prototypical implementations.

Programming language (mark): Which programming languages does the agent platform support for development and execution? Support for more languages yields higher marks. Nonetheless, we specified Java to be the minimal requirement.

Graphical Administration of Platform (mark): Does the platform provide a graphical administration tool?

Monitoring (mark): Does the agent platform provide a tool to observe agent migration and execution?

Debugging (mark): Is there an integrated debugging tool?

Rapid Application Development (RAD) (mark): Does the platform include a full-featured editor including wizards and help functions?

Deployment (mark): Does the platform include support for the deployment of agents?

Architecture (binary): Does the platform support established design principles in design, development, and operation of agent systems (e.g., InteRRaP described by [3])?

3.5 Characteristic Properties

Characteristic properties describe properties specifically targeted at agent platforms. To analyze the seven aspects of characteristic properties we relied on documentation and prototypical realization of agents.

Mobility (binary): This aspect investigates if the platform supports mobile agents. It is divided into several sub-aspects (e.g., network traffic, migration strategies), which should be taken into consideration for evaluation.

Standards (FIPA and MASIF) (binary): Does the platform observe these standardization efforts?

3.6 Knockout Criteria

To streamline the evaluation process and to focus evaluation efforts on promising agent platforms we will introduce knockout criteria. In Table 1 all criteria are listed, which eliminate a platform immediately from further evaluation. These knockout criteria are not fulfilled by a platform if at least one aspect is not supported. Note that not the criterion eliminates the platform, but even a single aspect. Obviously, for the evaluation of knockout criteria only binary evaluation is easily applicable.

Criterion	Aspect
Environment	Does the platform support the Windows and UNIX operation environments?
Environment	Does the platform provide any documentation?
Development	Does the development platform support JAVA for agent development?
Characteristic Properties	Does the platform provide support for mobile agents?
Availability	Is there a test (full functional, limited, student) version available?

Table 1: Knockout criteria and aspects.

3.7 Ranking of Criteria

Having applied the knockout criteria, the next step is to assign weights to criteria (see Table 2) so that the platforms can be ranked. This ranking establishes the sequence in which platforms are to be evaluated in greater detail. As the number of platforms that pass the knockout criteria may be arbitrary large we have to validate and prioritize other criteria to get an ordered list of agent development platforms according to our requirements.

If a criterion is valued by a binary, a value of '4' is assigned for 'No' whereas '1' is assigned for 'Yes'. If a feature cannot be evaluated it is graded as if not available (e.g., '4').

Criteria / Aspects	Weight
All aspects of security, e.g., authentication, encryption, certification, authorization, access restriction	50 %
Development state of the platform (alpha, beta, preview, release, technology study, academic)	20 %
Development supporting tools	25 %
Standards (FIPA and MASIF)	5 %

Table 2: Relative weights of criteria.

3.8 Test Specification

In this subsection we will show what additional specifications were required for a prototypical implementation, which constitutes in-depth analysis of the short listed platforms. In Table 3 the test environment is described, followed by an elaboration on how to assess mobility, the ease of administering

agents, network traffic analysis, and last but not least issues of stability.

Hardware	Description
Processor	Pentium III 600
Memory	256 MB RAM
Network	100 Mbit Ethernet

Table 3: Test system specification.

3.8.1 Mobility

For our prototype we will consider only weak migration, as Java does not provide support for storing an objects' current execution state and its serialization.

Regarding mobility, the prototype should be capable of (1) migrating to another platform, (2) looking autonomously for its path of future migration, and (3) returning to its host of origin.

The implementation of all agents will be based on following high-level design. The agent has an internal vector of the places to visit and keeps track of where it has already been. The next host to migrate to will be randomly selected. Once the internal list of the hosts to migrate to is empty, the agent should return to its origin.

3.8.2 Administration of Agents

After migrating to a new platform the agent should open a dialog window displaying its prior and next location. Finally, as soon as the user closes the dialog, the agents will migrate to the next platform. This allows getting an overview of the administration tasks and how tools can help developers or server administrators. The migration should be visible on the administration GUI and the agents' migration path can be followed easily.

3.8.3 Network Traffic

We want to test whether the delay introduced by migration only depends on network topology and how much bandwidth is actually required. As quantitative analysis proves to be tedious, we will evaluate this aspect in a qualitative way only. If a significant delay in migration between two computers is observed, we will have to find the reasons for this behavior. In the prototype we will measure delays by measuring the current system time just before migration; this time will be stored and carried by the agent while migrating. On the arrival at the next platform the first statement will measure the current system time. Departure time and arrival time are displayed in the message box mentioned above. Obviously, clocks have to be synchronized. During network performance tests we turned off all security features.

3.9 Stability

To evaluate this hard to measure aspect we will observe migration with regards to the following details.

(1) Does the agent platform delete the agents' data and threads after migration? This can be observed by tracking memory allocation of the processes.

(2) Does the platform remain in a stable state after frequent migration? This can be evaluated by continuously executing the prototype agent. If the platform is as responsive as it was before and if memory allocation of the platform did not increase depending on the number of iterations the platform is considered to be stable.

4. EVALUATION RESULTS

In this section we will first summarize our findings based on an evaluation of agent platforms performed in November 2000. Thereafter we will rank the platforms and will finally show which platforms have been select for further evaluation.

4.1 Knockout Criteria

A criterion can either be binary or evaluated by a mark. For a binary criterion the possible values are 'Y' (Yes) or 'N' (No). For marks the value '1' is best and '4' worst. In the case of ranking criterion 12 (Programming language) a value of '4' means that the platform supports the minimal requirement, namely Java. A value of '3' indicates that the platform supports a second language, a '2' corresponds to 3 supported languages, and a mark of '1' is awarded if more than 3 languages are being supported.

Note that a question mark in the matrix indicates a criterion that was not evaluated because it was no longer relevant (e.g., if the platform fails one knockout criterion others do not have to be evaluated).

The main focus of our evaluation lied upon security features of agent development platforms. The second most important issue includes development support of the platform. Finally, we judged conformity with the standardization efforts of FIPA and MASIF to be relevant, too. We are aware of the fact that standardization efforts are in a constant state of flux and are likely to be better supported in future.

It is evident that many platforms were eliminated due to knockout criteria such as support of mobility. However, this is not necessarily a deficiency as a considerable amount of platforms do not strive for mobility but instead focus on learning and planning capabilities. Moreover many platforms are still in their infancy or were never intended to be more than a feasibility study.

		KO-Criteria					Ranking													
		A[1]	E[2]			C[3]	Security					A[1]	Development							C[3]
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	Platform	Is there an evaluation version available?	Is the platform operating system independent?	Does the platform provide any documentation?	Does the development platform support Java?	Does the platform provide support for mobile agents?	Authentication	Encryption of data	Certificates	Authorization	Access Restriction	Platform state	Programming language	Graphical Administration of Platform	Monitoring	Debugging	Rapid Application Development	Deployment	Architecture	Standards (FIPA and MASIF)
1	Agentbuilder	Y	Y	Y	Y	N	Knocked out													
2	Aglets	Y	Y	Y	Y	Y	Y	3	4	1	1	3	4	1	3	4	4	1	Y	Y
3	Ajanta	Y	?	Y	Y	Y	Installation failed													
4	Bee-gent	Y	Y	Y	Y	Y	Y	2	4	4	4	3	4	4	4	4	3	4	N	Y
5	Bond	Y	Y	Y	Y	Y	Installation failed													
6	Concordia [4]	Y	Y	Y	Y	Y	Y	1	1	1	1	1	4	1	1	1	4	2	N	N
7	Cougaar	?	?	Y	Y	N	Knocked out													
8	D'Agents 2.0	N	Y	Y	Y	Y	Knocked out													
9	FarGo	Y	Y	Y	Y	Y	N	4	4	4	4	4	4	3	3	3	4	4	N	N
10	FIPA-OS	Y	Y	Y	Y	N	Knocked out													
11	Gossip 1.01	N	Y	Y	Y	Y	Knocked out													
12	Grasshopper	Y	Y	Y	Y	Y	Y	2	2	2	2	1	4	2	2	4	4	1	Y	Y
13	Gypsy	Y	Y	Y	Y	Y	Installation failed													
14	Hive	Y	Y	Y	Y	Y	N	4	4	4	4	3	4	1	1	4	4	3	N	N
15	IBM-Able	Y	Y	Y	Y	N	Knocked out													
16	JACK	Y	Y	Y	N	N	Knocked out													
17	Jade	Y	Y	Y	Y	Y	N	4	4	4	4	1	4	2	3	3	4	2	N	Y
18	Jafmas	Y	Y	Y	Y	N	Knocked out													
19	Jess	?	?	?	?	N	Knocked out													
20	Jumping Beans	Y	Y	Y	Y	Y	Y	1	3	1	1	1	4	2	4	4	4	2	N	N
21	Kaariboga	Y	Y	Y	Y	Y	N	4	4	4	3	4	4	3	4	4	4	2	N	N
22	MadKit	?	Y	Y	Y	N	Knocked out													
23	Mast	Y	Y	Y	N	Y	Knocked out													
24	Mole	Y	Y	Y	Y	Y	N	4	4	4	4	1	4	3	4	4	4	4	N	N
25	OAA 2.0.9	Y	Y	Y	Y	N	Knocked out													
26	Pathwalker 1.0	Y	Y	Y	Y	Y	N	4	4	4	4	3	4	4	4	4	4	4	N	N
27	Ronin 1.1	Y	Y	Y	Y	N	Knocked out													
28	Soma 2.0/3.0 Beta [5]	Y	Y	Y	Y	Y	Y	1	2	2	4	1	4	1	2	4	4	1	Y	Y
29	Voyager 3.	Y	Y	Y	Y	Y	Y	1	4	4	4	1	4	4	4	4	4	4	N	N
30	Zeus 1.02	Y	Y	Y	Y	N	Knocked out													

1) Availability

2) Environment

3) Characteristic Properties

4) Concordia could not be evaluated because of export restrictions!

5) During evaluation only italian documentation was available - english documentation will be available soon!

Table 4: Overall criteria evaluation matrix.

4.2 Ranking

Based on the above findings we proceed by assigning weights to criteria so that we can compare all platforms. First we have to map binary information to numeric values so that they are comparable to marked criteria. For binary criteria we assign '4' for 'No' and '1' for 'Yes'. If a feature cannot be evaluated it is considered to be not available. We can now rank platforms using the weights as previously defined.

Table 5 shows the result of the criteria weighting. Smaller values suggest that a platform fulfills our requirements better and should therefore be considered for further evaluation.

	Security						Development								Result
	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
Grasshopper	1	2	2	2	2	1	4	2	2	4	4	1	1	1	9,25
Jumping Beans	1	1	3	1	1	1	4	2	4	4	4	2	4	4	9,9
Aglets	1	3	4	1	1	3	4	1	3	4	4	1	1	1	10,15
Voyager 3.3	1	1	4	4	4	1	4	4	4	4	4	4	4	4	14,4
Bee-gent	1	2	4	4	4	3	4	4	4	4	3	4	4	1	14,9
Jade	4	4	4	4	4	1	4	2	3	3	4	2	4	1	15,75
Hive	4	4	4	4	4	3	4	1	1	4	4	3	4	4	16,05
Kaariboga	4	4	4	4	3	4	4	3	4	4	4	2	4	4	16,75
Mole	4	4	4	4	4	1	4	3	4	4	4	4	4	4	17,15
FarGo	4	4	4	4	4	4	4	3	3	3	4	4	4	4	17,25
Pathwalker 1.0	4	4	4	4	4	3	4	4	4	4	4	4	4	4	17,8

Table 5: Application of the criteria weighting.

4.3 In Depth Evaluation

While evaluating platforms we found that nearly each one has its own philosophy regarding development of agents. The standardization efforts undertaken by FIPA and MASIF are considered only by a few agent development platforms. Therefore substituting one platform by another requires a redesign of all the code. However, it seems probable that an increasing number of platforms will follow the standard architectures in near future.

As we were not able to download the Concordia platform due to export restrictions and because Soma 3.0 Beta did not provide an English programmer's guide, we decided to exemplarily evaluate and test the Grasshopper 2.1 platform [2] in depth. For a detailed report please refer to [8].

Grasshopper is available both for Windows and UNIX version as it is based on pure Java. A user's guide - though not fully complete, a programmer's guide and some presentations are available at the Web site. JavaDoc API documentation is included with the distribution. Support exceeding bulletin boards and FAQs is available for a fee.

As Grasshopper relies entirely on Java, agents may only be developed in Java. Support for mobile agents is integrated in its architecture.

Moreover, Grasshopper supports internal and external security mechanisms. For internal security role based authentication and for external security PKI is used. Grasshopper's role-based authorization system is based on policy files located on every computer but currently offers no wizard supporting deployment of these policy files. Additionally, Grasshopper secures agents in migration based on SSL and also supports X.509 standard digital certificates.

Grasshopper is delivered with a graphical administration tool based on Swing and also has a text oriented interface. On starting Grasshopper one can decide which administration to use.

There are also several drawbacks such as that integrated debugging is not supported and that there are no development wizards available. Furthermore, despite being simple the architecture of agents is not standardized or transparent. Nonetheless, Grasshopper follows FIPA and MASIF by offering a separate extension available at the Web site.

Finally, we want to emphasize that Grasshopper is in production state and well supported by mailing list as well as commercial support sources.

5. CONCLUSION

As the area of agent development platforms is in a constant state of flux any comparison published in print is inevitably at least partially outdated. Our contribution, however, strives to provide readers with a framework that is easy to adapt to specific requirements when confronted with the challenge of selecting an agent development platform for industry-scale projects.

During and after the evaluation of agent platforms, we already isolated real-world problems that could be solved by the use of mobile agent technology. The first problem is the maintenance of schemata and the execution of queries in a heterogeneous distributed database environment. In this area security, performance, and administration effort plays an important role. Our approach is to make SQL statements and procedures mobile, send them directly into the databases, and let them perform their tasks locally. In this case, the agent platform Grasshopper is used to transport the SQL statements as well as

procedures to their destination databases and to handle route planning as well as security issues.

A second problem that could be solved by the application of a mobile agent platform is automatic software distribution across the Internet. Existing software distribution products provide insufficient support of automatic online distribution across the Internet, are limited in error handling, and provide insufficient support for mobile devices as well as inadequate interoperability of different products. An agent-based solution would be more flexible, more extendible, platform independent, and would provide higher interoperability between agent-based software distribution systems in order to support monitoring, decrease network traffic, allow central administration, and decrease maintenance work.

Mobile agents are an ongoing field of study within the ever-growing autonomous agents arena. The qualitative strengths of mobile agents, such as their support for disconnected operation and dynamic deployment, make current agent platforms an attractive choice for a wider and wider range of distributed applications.

6. ACKNOWLEDGMENTS

This work has been done in the framework of the Kplus Competence Center Program, which is funded by the Austrian Government, the Province of Upper Austria, and the Chamber of Commerce of Upper Austria.

7. REFERENCES

[1] [Gray, 2000] Gray R. S., Cybenko G., Kotz D., Rus D., Mobile agents: Motivations and State of the Art, in Jeffrey Bradshaw (Eds.) Handbook of Agent Technology, AAAI/MIT Press, 2000.

[2] [IKV, 1999] IKV++ GmbH, Grasshopper Release 2.1 Programmer's Guide, http://www.grasshopper.de, July 2000.

[3] [Müller, 1994] Müller, Pischel, Thiel, A pragmatic approach to modeling autonomous interacting systems, in Wooldridge, Jennings (Eds.) Pre-Proceedings of the 1994 Workshop on Agent Theories, Architectures and Languages, 226 - 240, Amsterdam, The Netherlands, 1994.

[4] [Ghezzi, 1997] C. Ghezzi, G. Vigna, Mobile Code Paradigms and Technologies: A Case Study, in Proc. Mobile Agents `97, Springer Verlag, 1997.

[5] [SCCHWeb] Webpage of the EvalAgents Project, http://www.scch.at/research/projects/EvalAgents/home

[6] [Sierra, 2000] Sierra, Wooldridge, Adeh, Agent Research and Development in Europe, IEEE Internet Computing, 81-83, Sep/Oct 2000.

[7] [TR0049, 2000] Altmann, Grabner, Gruber, Klug, Stockner, Agent Technology: State of the Art, Technical Report 49/2000, Software Competence Center Hagenberg, 2000.

[8] [TR0064, 2000] Altmann, Grabner, Gruber, Klug, Stockner Evaluation of Agent Platforms, Technical Report 0064/2000, Software Competence Center Hagenberg, 2000.

[9] [UMBC-AgentWeb] UMBC AgentWebpage. http://agents.umbc.edu/, University of Maryland, Baltimore County.