

Making a Strong Business Case for Multiagent Technology

Steve S. Benfield
CTO, Agentis Software
1200 Ashwood Parkway
Suite 410, Atlanta, GA 30338
+1 678.368.4105

sbenfield@
agentissoftware.com

Jim Hendrickson
VP Global Services, Agentis Software
1200 Ashwood Parkway
Suite 410, Atlanta, GA 30338
+1 650.759.8995

jhendrickson@
agentissoftware.com

Daniel Galanti
Dir of Engineering, Agentis Software
Reconquista 865 - Piso 6
Buenos Aires, Argentina
+54.11.4317.6880

dgalanti@
agentissoftware.com

ABSTRACT

In this paper, we describe lessons learned in commercializing and implementing agent-based systems in real-world business situations. Most notably we discuss gains in programmer productivity achieved by using AdaptivEnterprise, a commercial IDE and J2EE runtime engine based on BDI agents. In a wide range of complex business applications, we show that the use of BDI technology incorporated within an enterprise-level architecture can improve overall developer productivity by an average 350%. For java coding alone, the increase in productivity was over 500%. In competitive bids for large scale enterprise applications, AdaptivEnterprise was able to reduce total cost of ownership between two- and three-fold. In addition, the agent approach allowed the business to change and extend solutions quickly and helped bridge the gap between the business side and IT development.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence – *multiagent systems*.

General Terms

Management, Economics

Keywords

Agentis, AdaptivEnterprise, Function Point Analysis

1. INTRODUCTION

Agentis Software offers a commercial environment and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

COPYRIGHT 2006 ACM 1-59593-303-4/06/0005...\$5.00.

methodology, called AdaptivEnterprise, for implementing agent-oriented goal-seeking applications. To date, Agentis has either implemented or assisted customers in implementing 15 applications. Of these, 13 are currently in production and two are pending deployment.

AdaptivEnterprise is both an IDE for building agents as well as a Java runtime engine for deploying the agents in a scalable, reliable, and secure manner. AdaptivEnterprise deploys agents in a pure J2EE-environment and has been commercially available for several years.

2. MAJOR LESSONS LEARNED

Agentis has learned several major lessons in commercializing agent technology. They are:

- Agent technology is applicable to a wide range of industry problems;
- The primary business value of agent technology is reduced complexity;
- Agent technology provides compelling ROI and productivity improvement;
- Designing with agents is easier for business analysts than conventional technologies;
- An iterative methodology is critical to building successful Agent-Oriented Systems; and
- IT Programmers can easily make the transition from pure Object-Orientation to Agent-Orientation

3. AGENT TECHNOLOGY IS APPLICABLE TO A WIDE RANGE OF INDUSTRY PROBLEMS

Our solution, AdaptivEnterprise, is currently in use in a wide variety of applications across a range of industries including logistics, finance, energy, telephony, transportation, and housing administration.

The range of implemented applications includes:

- Datacenter compliance and monitoring (Sarbanes-Oxley Compliance)

- Multi-jurisdictional tax calculations
- Global package tracking
- Housing allocation
- Service configuration
- Dynamic pricing
- Customs documentation cleansing and classification
- Document workflow

4. REDUCING COMPLEXITY: THE #1 VALUE OF AGENTS

The most common theme throughout these applications is that they are “hard” to build. All are complex and are not the typical transactional applications that corporate IT groups normally build. Many of these applications are more typically delivered as customizable packages by third-party vendors. However, these applications have complex requirements and enough “exceptions” to make implementing an off-the-shelf package unfeasible without significant customization efforts.

Another common theme of these applications is change. Requirements for these applications change very rapidly and in often complex ways. New services and pricing schemes are added, companies are merged, government regulations change, etc. The ability to be able to quickly add or change functionality is critical. Because major control logic is abstracted in an agent-based system, adding and changing functionality is very easy.

These applications are also event and exception driven. That is, the system needs to respond to ongoing events and exceptions. Because the response to an event can vary depending on the current state of the system, the loosely coupled nature of agent control logic (processes) makes event and exception processing much simpler to code.

Finally, speed to market has been critical in the applications we’ve been involved in. Given the mission critical nature of these projects, there are serious financial and business implications involved in getting these applications completed. The reason many of our customers adopted Agent technology is that they were looking for a faster and easier way to get these applications built. Some projects were awarded to Agentis because we promised to deliver the application quickly, not because the customer understood Agent-based systems.

5. MEASURING ROI FOR PROJECTS

While the technical reasons for using agents might be apparent for various situations, in business settings, Return on Investment (ROI) is a primary decision criterion. Businesses will take a chance on a technology through a proof-of-concept project (POC) without regard for ROI but, for real implementations, the ROI must be compelling. Agent technologies are not pervasive in businesses and are foreign to many IT departments. In addition, there is a good bit of confusion about what agents are.

Measuring true ROI is complicated—and can be expensive to measure in full detail. In addition, it is sometimes hard to isolate the ROI of agent technology itself because many factors go into the success and the total cost of a project, including software licenses, analysis and design, testing, GUI construction, server infrastructure, project management, consulting/labor costs, etc.

Likewise, one cannot compare the ROI of two projects and assume the one with the lowest ROI was the most successful. Projects differ in complexity, internal politics, purpose, stakeholders and, of course, core functionality. Even two projects with similar functionality can have vastly different ROIs because of factors external to the products/languages used, such as changing requirements, ineffective management, etc.

However imprecise ROI measurements tend to be, there are several generally accepted measures of a project’s “success”. The first is productivity—how productive was the IT staff at implementing the solution versus other technology choices or development teams? The second is cost—how much did the project cost versus alternative estimates?

Given the fact that the applications that agents are best at solving tend to be complex critical applications, the expected ROI should be high. One specific project resulted in a 173% ROI and is detailed at the end of this paper.

5.1 Estimating Agent Productivity with Function Point Analysis

Function Point Analysis (FPA) is a technique published in 1979 by Allan J. Albrecht to measure the “size” of a software application and predict the level of effort associated with building it.

A Function Point (FP), according to the International Function Point Users Group (IFPUG.org), is a unit of “system functionality” measured from a functional, or user, point of view. It is independent of the computer language, development methodology, technology or capability of the project team used to develop the application.

According to IFPUG:

- FPA can be used to size software applications accurately. Sizing is an important component in determining productivity (outputs/inputs).
- Function Points can be counted by different people, at different times, to obtain the same measure within a reasonable margin of error.
- Function Points can be used to determine whether a tool, a language, or an environment is more productive when compared with others.

Because FPA is a fairly objective methodology for analyzing software complexity, it is useful for measuring

the productivity of differing software teams, methodologies, frameworks, and languages.

One of Agentis' customers, a large international logistics firm, uses FPA as a standard for measuring projects and IT productivity. Recently, they performed an analysis of their Agent-Oriented projects to determine whether to increase the role of Agents in the organization.

Based on six applications encompassing a total of 7,356 function points (FP) and 10 person-years of development, AdaptivEnterprise delivered an average of 2.11 function points per day for each fully completed project. Average productivity for all other non-agent-based Java projects within the organization is 0.45 FP per day.

Based on this analysis, agent-oriented methodology, architecture, and development delivered a 368% improvement in overall project productivity. (2.11 FP Actual/0.45 FP Expected)

Improvements on specific projects ranged from a high of 513% improvement to a low of 273% improvement. The largest single project had 3,850 FP and the smallest project had 304 FP.

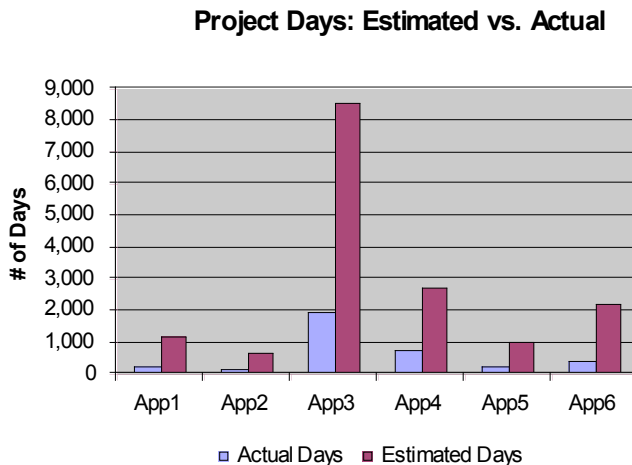


Figure 1: Estimated vs. Actual Project Days

5.2 Agent-Orientation's Effect on Coding

All the FP numbers presented to this point have been for the complete project—which includes requirements gathering, analysis and design, testing, and implementation. When Agent-oriented development is compared at a pure coding level, the productivity improvements are even greater.

Since AdaptivEnterprise is a 100% pure Java and J2EE solution, we compared the agent-based applications against other Java applications in the organizations. The historical average FP coding productivity for the organization's Java developers is 0.95 FP/day.

We only have coding metrics for four of the projects and their mean coding productivity with AdaptivEnterprise was

4.97FP/day. This represents a 423% improvement (4.97FP / 0.95 Average Java FP) over the standard for the organization.

Finally, a non-FP related data point is provided by an application for another customer involving a migration of an existing COBOL workflow application with 120,000 lines of code. The application was re-engineered into a Java-based system with seven agents in less than three months with two full-time and one part-time developer.

6. DESIGNING WITH AGENTS IS EASIER FOR BUSINESS ANALYSTS

How do you take a business process and map it into software? This is a central problem that business is still striving to solve in application development. Translating requirements into a working system is as much a science as it is an art. Humans who work with business processes speak one language while programmers and systems analysts speak another. IT staff are much more concerned with wiring specific systems and functions together to form a specific process. Business analysts describe roles, patterns, workflows, and decision trees.

We have found that by discussing systems in terms of data, goals, and plans, the conceptual modeling of systems becomes much easier. First, the terminology is easier for both IT and business to understand. Agent analysis lends itself very well to iterative requirements gathering. Just as it is easy to code new plans to an existing Agent-based application, it is also easier to add new plans during the conceptual phase of an application. Likewise, moving from the conceptual phase of analysis to the concrete design phase is aided because agent-oriented analysis produces goals and plans that can then be specifically designed and coded as discrete work elements.

At Agentis, we've used a modified version of the Rational Unified Process along with agile development techniques such as Scrum or XP. As pointed out previously, time to market is important with these applications. Because of that, a classic waterfall or even a strict RUP-based UML modeling approach is not always possible or desirable for these projects.

6.1 Agility Is Critical

A typical situation that we encountered was a rapid escalation in the problem space. Many organizations that are introduced to Agents for the first time are a bit skeptical. To mitigate any possible risks involved with bringing Agent-technology on board, many projects start as smaller proofs-of-concept and are quickly escalated into full-blown formal projects once Agent technology and concepts are proven.

Besides being able to tackle complex problems, Agentis' experience is also one of agility. Agent-based approaches allow us to respond very quickly to changes in the

requirements of a system and to changes to systems currently in production. For example, the application that covers 3,850 function points started life as a 350+ function point application. The benefits of the system were immediately seen by the customer and the application quickly escalated into a major, mission-critical endeavor. Because the BDI agent-model scales easily, it was fairly straightforward to move the system from a small effort to a large one—more agents were added along with more goals and more plans.

Several customers have stated that agent-oriented methodology coupled with a GUI design environment for building agents facilitated RAD capabilities with their complex systems—something that was previously not thought possible, since RAD tends to be associated with smaller systems with more CRUD (Create, Read, Update, Delete) transaction functionality.

6.2 Agent Technology Must Play Well With Others

Both the methodology as well as the actual implementation of an agent-based system must “play well with others”. At minimum, a system will need to interact with an existing relational (SQL) database—most likely by storing long running state & transaction information.

Many organizations are moving toward Service Oriented Architecture (SOA) by implementing standard language and platform neutral mechanisms for exposing computing resources through WSDL or SOAP. At minimum, the agent system must be able to consume SOAP-based web services and provide agent-based web services to other systems via web services standards.

Likewise, organizations are also starting to use BPEL as a process model definition and execution environment. Agents must be able to call BPEL processes and BPEL processes must be able to invoke agents.

Having an Agent technology based on J2EE or .NET mitigates most of these integration concerns at a code level. Having GUI and IDE support for these interactions is critical for productivity.

6.3 Agent Technology Can Be Used By IT Programmers

Agentis normally helps customers build their first application. In cases of large applications, a third-party integrator might be brought in to provide programming resources. Projects are normally led by Agentis staff with extensive agent experience. However, most of the coding is done by “normal” Java developers. Agentis has trained over 100 developers in the AdaptivEnterprise product and all of our customers have taken over the maintenance of their applications with several of them continuing to build new applications with no direct input from Agentis.

Typical developer training is three days of classroom training on the product and 1½ days on the methodology. We have added concept courses for the Business Analyst and Architects. Developers can be productive after the training and proficient in 4 to 6 weeks of project work.

7. CASE STUDIES

7.1 Multi-Jurisdiction Tax Application for a Large Energy Company

Collection of sales and use tax is complicated because each city, municipality, county, and state has their own taxation policies. In addition, many large corporations are given tax breaks and have specific agreements in place with various government agencies covering the types of items that are taxable and at which special rate. Finally, because the customer in this case is an energy utility, many special taxation rules apply that are above and beyond typical use tax calculations.

The customer had acquired an existing tax package and had spent almost 18 months of time and several millions of dollars implementing and customizing the application. However, the tax package was unable to provide accurate tax information because of the complexities involved and required heavy user intervention, resulting in large amounts of overtime pay spent to ensure the tax calculations were correct. Agentis was brought in to help fix the problem.

The customer was attracted to the speed of development that Agentis promised as well as the goal and plan nature of the agent architecture which was easy for the customer to understand. Likewise, speed of application maintenance and change was critical. Tax laws and regulations change constantly across many tax jurisdictions. The agent-oriented application allows them to add and modify tax calculation plans very easily.

Between overtime spending, maintenance spending on the discontinued tax application, and ability to meet quality of service requirements of the government, the customer saved just under \$6 million in direct costs and administrative revenue for collecting & administering the taxes.

ROI for Tax Application	
Direct Savings & Revenue Stream	\$5,812,000
Project Costs	\$ 1,171,000
Useful Life	5 Years
Net Present Value	\$ 2,005,555
ROI	173.06%

Overall the ROI on the project was calculated at 173% over five years resulting in an internal rate of return of 43%. In

addition to the cost savings, the customer can now offer value-added services to their customers such as tax reduction strategies, etc. In addition, the new system provides an audit trail for additional Sarbanes-Oxley (SOX) compliance.

The project was so successful that the customer has contacted Agentis on how the two companies can become closer strategic partners for further application development.

7.2 Global Package Tracking, Tracing, and Recovery

The customer is a global logistics and shipping company handling billions of packages a year. As the organization grows and acquires other logistics firms, a global solution was needed to handle package tracking, tracing, and recovery. As packages move through the system, millions of events are generated which change the state of the system and of individual shipments.

The application was designed to handle up to 7,500

a shipment or entering details about the shipment and its current status: examples include being detained for customs, weather delays, transit strikes, etc. Because of this, human intervention is usually required in the process as well as timely communication to the customer on the shipment status. To support this complex workflow scenario, the system supports the following:

- Role-based and team-based workflow;
- Integration to existing enterprise services for authorization and authentication, shipment searching, customer location and service level agreements;
- Real-time subscription to changes in physical location and workflow state;
- Automatic workflow task allocation to most suitable Customer Service Representative (CSR) and most suitable team; and
- Extensive workflow monitoring: notifications, alerts, reminders, state change markers.

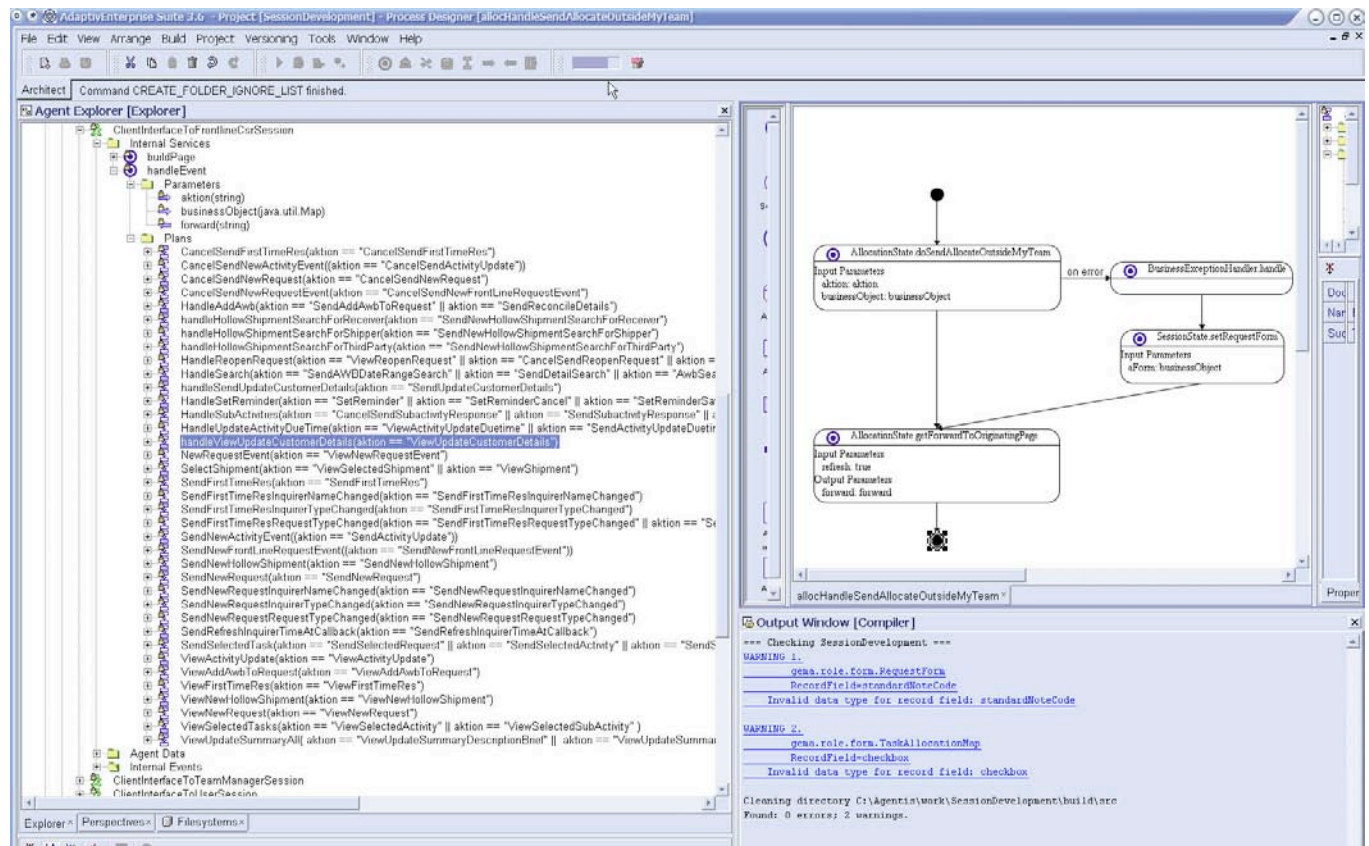


Figure 2: An event-handling agent with 37 possible executable plans.

concurrent users over 100 countries and needed to handle over 10,000 traces per day. A trace is a physical trace of a shipment—not just a check on ‘where is my package’. Many times a trace results in people physically looking for

The system is currently deployed to over 30 CPUs serving almost 3,000 concurrent users.

Business benefits, as stated by the customer, are:

- Reduced cost of operations because of automation of tasks and increased call center throughput;
- Enterprise-level adaptability because of agent reuse and flexibility and SOA implementation; and
- New platform for customer self-service as well as new service delivery channels

The reasons the customer chose Agentis/Agent Technology were:

- Full BDI/Goal Directed Agent Framework
 - 100% J2EE solution; use of EJBs, MDBs, JMS
 - Both synchronous and asynchronous inter-agent communication
- Extreme adaptability through Agents
 - Low lifecycle costs: maintenance is adaptation
 - Unprecedented application longevity
- Agents are EJBs
 - Enterprise friendly and very scalable
- Visual Agent Development Environment
 - Oriented toward Business Analysts
 - Comprehensive develop/test/deploy environment

The system was competitively bid with the customer. The total implementation costs to date are roughly \$6MM. Other competitive solutions using a combination of packaged software along with customization had an estimated total ownership cost of \$15MM to \$25MM. The customer was able to get the functionality that they wanted in half the time they wanted at less than half the cost.

8. Summary

Agent technology is ready for mainstream corporate use and can be successfully commercialized. It has proven itself to be highly reliable and scalable and also well suited for complex business applications involving many moving parts, fast changing requirements, and tight timelines. Agent-based solutions have been used many times to solve problems customers thought were not solvable—at least not within any reasonable timeframe or budget constraints.

However, pure agent technology is not enough. The technology must be presented in a way that “typical” programmers can understand and use—and it must integrate into their current development efforts.

Likewise, a proven methodology that addresses the needs of business analysts as well as developers is also required.

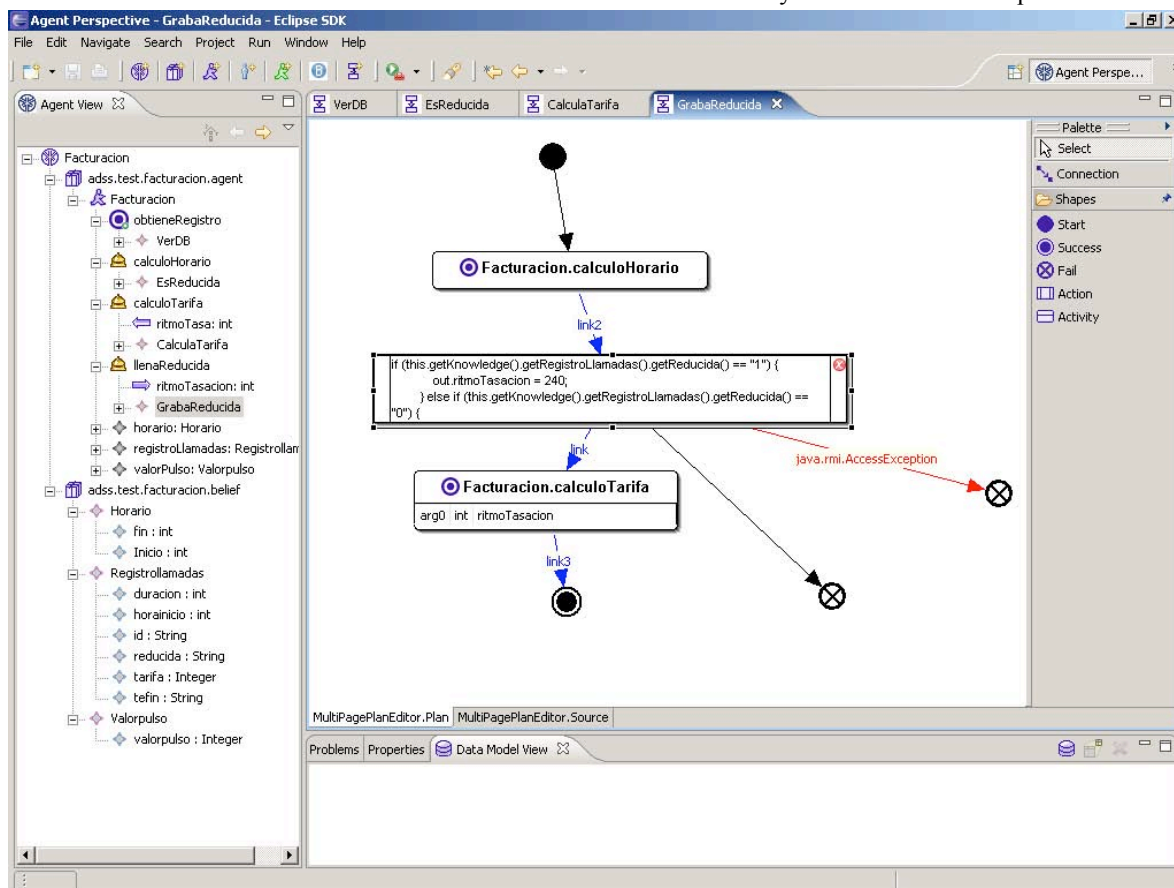


Figure 3: Plans are modeled with UML.