

Manufacturing Systems Integration and Agility: Can Mobile Agents Help?

Authors

Todd Papaioannou and John Edwards
Department of Manufacturing Engineering
Loughborough University
Loughborough
Leicestershire, LE11 3TU
UK
Tel.: +44 1509 228250
Fax.: +44 1509 267723
Email:
T.Papaioannou@lboro.ac.uk
J.Edwards@lboro.ac.uk

ABSTRACT

Mobile code is being championed as a solution to a plethora of software problems. This paper investigates whether Mobile Agents and Mobile Objects support improved system integration and agility in the manufacturing domain.

We describe two systems built to support the Sales Order Process of a distributed manufacturing enterprise, using IBM's Aglets Software Development Kit. The Sales Order Process model and the requirements for agility used as the basis for these implementations are derived from data collected in an industrial case study.

Both systems are evaluated using the Goal/Question/Metric methodology. Two new metrics for Semantic Alignment and Change Capability are presented and used to evaluate each system with respect to the degree of system agility supported.

The work described provides evidence that both Mobile Agent and Mobile Object systems have inherent properties that can be used to build agile distributed systems. Further, Mobile Agents with their additional autonomy provide marginally greater support.

Keywords Mobile Agents, Mobile Objects, Enterprise Integration, Distributed Systems, System Agility, Sales/Order Processing, Aglets.

1 INTRODUCTION

Access to an organisation's information is a critical factor in developing business systems. Regardless of the architecture employed, the information stored in an organisation's databases is its lifeblood. The ability to effectively manage, manipulate, and distribute this information was once viewed as a provider of competitive advantage (White 1994). Today it is simply a base requirement for corporate survival within the global market place.

Increasingly, the operations of a manufacturing enterprise are being distributed geographically, and thus the supporting information technology (IT) systems must themselves be capable of distributed operation (SSA 1995). Management of information remains a base requirement, as does the integration of differing IT systems and data sources.

Internet technology has proved to be effective for connecting a mix of different types of computers and computer networks, while also providing location independence to information. Further, analysts predict that the near future will see the evolutionary convergence of the Internet, Intranets and traditional IT models such as client/server and peer to peer (SSA 1995). However there remains a number of problems with this technology.

The saturation of network bandwidth, especially when part of the network in question is the Internet, means that remote database access as required by the distributed enterprise model could mean ineffective IT support for the business. As well as data timeliness, factors such as data integrity and security are also a concern when dealing with the Internet.

Mobile agent technology has been proposed as a general solution that has the potential to overcome this set of problems. It achieves this through local interaction (Clements et. al. 1997) and is equally applicable to the problem of geographically distributed information sources, since mobile agent systems are inherently distributed. Both mobile objects and mobile agents are used in our work to build and integrate software systems that support distributed manufacturing enterprises, in order to test whether mobility can help. Evaluation of the two systems is conducted using Basillii's GQM methodology (Solingen and Berghout 1999), through which a number of relevant metrics are identified. The following section describes the model used in this work.

2 THE MODEL

The systems described in this paper are implementations of a real-world Sales Order Process (SOP), typical of many manufacturing enterprises. The model upon which this system is based has been derived from an industrial case study undertaken at a high-pressure vacuum component manufacture that is based in the UK (from herein referred to as Acme).

Acme currently uses a mixture of bespoke packages and standard Office products to support its business. However, as the needs of the market change ever more rapidly the frailties in the existing IT infrastructure become increasingly apparent. To survive, Acme must remain agile and competitive, and to achieve this their IT infrastructure must be capable of responding to change. This requirement is not being met by the existing IT infrastructure. For example, in a recent experience at Acme it took two and half years to phase out an old accounting package and introduce the new software; clearly this is not acceptable. Equally, the strategy of buying inflexible, monolithic “off the shelf” packages is beginning to impact on the business, as these rapidly become inflexible legacy applications. The intricate spider's web of interdependencies woven between those packages that are already in place is making it increasingly difficult to contemplate radically changing the supporting infrastructure. As the interdependencies increase, the agility of the company is jeopardised. This legacy IT problem is now well recognised within many application domains.

The overview of Acme's business processes generated from this study can be seen in Figure 1. In this diagram, the separate business processes involved in the SOP are defined by the senior management figures that are responsible for those particular areas. Each core process is surrounded by a dotted line for further clarification.

From this view, it is possible to identify and extract the core business processes and represent them in a higher level abstract view. By examining the interactions between the major processes, a simple top level process model was generated to represent the entire SOP within the Acme business. Figure 2 shows a representation of the top level business processes model and a “walk through” description of its elements and their interactions.

The model clearly shows the interaction of a number of processes which are supported by specific sub systems, demonstrating the importance of system integration in this domain. In

order to evaluate whether mobility can help manufacturing enterprise integration we must first examine the integration methods currently employed.

3 TRADITIONAL ENTERPRISE INTEGRATION

As the use of computing in business has become increasingly fundamental to the success of an enterprise, so the requirements for what is achievable from this technology have increased. Global businesses require an infrastructure that is capable of dealing with distributed systems, made up of constituent components, that are able to communicate over different hosts, perhaps even different domains on different sides of the world. A number of infrastructural models have been proposed, and many are in use today, typically Common Object Request Broker Architecture (CORBA) (OMG 1994), the Distributed Common Object Model (DCOM) and Remote Method Invocation (RMI) (Sun Microsystems 1998). All of these technologies attempt to solve the distribution problem by providing location transparency, a major tenet of all distributed systems in the Reference Model for Open Distributed Processing (RM-ODP) (ISO 1992) family. The authors contend that the principal of local interaction as embodied in the mobile code paradigm provides potential for improved integration and agility when compared with the traditional technologies. The following subsection summarises some of the characteristics of CORBA in order to establish a context for examining the potential advantages of mobility.

3.1 CORBA

CORBA is the product of a consortium of over seven hundred companies, known as the Object Management Group (OMG). The OMG has approached the problem of integrating disparate components by creating interface specifications, and not code. Distributed components of the system are able to describe their interfaces using the Interface Definition Language (IDL) and subsequently interoperate through the underlying Object Request Broker (ORB) (Orfali et. al. 1996). The ORB functions as the communication backbone through which all of the system components are able to interact. The OMG contend that components communicating via an ORB do not need to be aware of the mechanisms used in that communication, and in fact are able to discover each other at run time allowing flexibility and configurability. However, the reality is that in the currently available commercial ORB's all that can be discovered at runtime are a component's methods and arguments as described by its IDL definition. This provides insufficient support for

creating tangible solutions based on loose coupling and late binding (Coutts and Edwards 1998). It is possible that the addition of available Trading Services, as described in the OMG model, could provide improved levels of support for this type of solution.

CORBA brings the location transparency abstraction to the integration issue, and whilst this is a useful metaphor for distributed systems, it does have drawbacks (Waldo et. al. 1994). In Figure 3, we represent two components communicating via an ORB. Although it appears to Component A that it is interacting with Component B on the same system, this may not be the case. Component B could be on a completely different host or network, or even on the other side of the world.

3.2 Summary

Traditional distribution mechanisms promote location transparency as one of their major advantages, i.e. they achieve communication between different objects by “hiding” the location of these objects from each other. Their messaging systems allow mobile data to be passed between objects and locations. Thus, traditional systems can be characterised as using location transparency and mobile data to integrate and enable communication between distributed objects. This characterisation is shown in the top part of Figure 4.

It is the authors’ contention that the features of local interaction, reduced network usage, server flexibility and application autonomy, which are supported by mobile agent technology, have the potential to provide a level of integration and agility above that provided by more conventional IT technology as embodied by CORBA (Papaioannou and Edwards 1999).

4 MOBILITY: AGENTS and OBJECTS

Mobile Agents have been a burgeoning topic for discussion in mainstream agent based research for several years. Mobility in general has an expanding community dedicated to investigating the potential of this new paradigm (MML). Although there is still no definitive answer as to whether mobile code systems will live up to the expectations placed upon them, there is a growing body of work (Vitec and Tschudin 1997, Rothermel and Hohl 1998, Papaioannou and Minar 1999) and an expanding number of frameworks, including Aglets (Lange and Oshima 1998), Voyager (Object Space 1997), Mole (Straber et. al. 1996) and Hive (Minar et. al. 1999).

4.1 Classification

The mobile agents discussed and used in the work described in this paper will be defined as:

“Software agents that are able to autonomously migrate from one host to another in a computer network.”

They can also be classified in line with Franklin and Graesser's agent classification scheme (Franklin and Graesser 1996) as *goal oriented*, *communicative*, and *mobile* i.e.:

- Goal oriented - they are not simply reactive but proactive
- Communicative - they are able to communicate with other agents
- Mobile - they can transport themselves from one host to another

The mobile objects discussed and used in the work described in this paper are similar to the agents described above, the key difference being that they do not contain the logic that enables the agents to make autonomous decisions regarding sales order processing.

4.2 Background

By their very nature, mobile objects and agents are inherently distributed. As such, they must be executable across a variety of platforms and operating systems to achieve their full potential. In a closed, controlled, network there may only be one configuration upon which they must work, but their true advantage comes from being able to migrate to disparate systems and continue functioning. This requirement has influenced the way in which mobile code systems are created and it is usual for mobile agent frameworks to be written in some type of script or bytecode that can be interpreted, usually in a dedicated host or server. Indeed, the spiralling popularity of Java, combined with its platform independence, has made it the de facto language for mobile code systems. Script or bytecode interpretation removes the need to re-compile an agent on arrival at a new host, instead placing the onus on ensuring an environment exists at the host that is capable of uniformly executing the agent on arrival. Most examples of these systems have a server or some type of executing environment in which the agents are hosted (Lange and Oshima 1999, Gray 1997). An overview and discussion of Java mobile agents can be found in (Kiniry and Zimmerman 1997).

4.3 Advantages of Mobile Code Systems

There have been many advantages claimed for mobile agents (Chess et. al. 1994, Lange and Oshima 1999). Unfortunately, very few quantitative measures exist to support these claims. However, a summary of some of the more frequently quoted qualitative ones include (MML):

Bandwidth savings

Instead of transferring lots of data back and forth over the network, move the logic to be local to the data source.

Limiting latency

By having the logic and data on the same machine, latency can be reduced.

Disconnected operation

If a computer is going to be offline at times, then it is possible to move the active processes to another host.

Stability

By using mobile code, software can be less dependent on the network, and therefore more stable. Mobility can be used to achieve replication for fault tolerance.

Server Flexibility

A client isn't limited to the functions a server pre-defines. Code mobility allows clients to upload new (or improved) functionality to the server.

Simplicity of installed server base

Servers can become simpler, requiring less functionality pre-engineered from the outset. This can help with preventing legacy.

Enable distributed computation

Mobile agents are inherently distributed, and as such can be a fundamental enabler for distributed computation.

4.4 Summary

Mobile code systems are quite different to existing traditional distribution systems prevalent in industry today. We characterise mobile object systems as providing local interaction for communicating objects, and providing an abstraction based on mobile messengers with limited autonomy as shown in the central section of Figure 4. We characterise mobile agent systems as providing local interaction for communication, through an abstraction based upon mobile logic *and* data. In addition, mobile agents provide greater autonomy. This characterisation is shown in the bottom section of Figure 4. Figure 5 depicts the notion of a mobile agent migrating from Host A to Host B so that it can reside locally to a data source it wishes to query.

5 INDUSTRIAL MOTIVATION

The case study at Acme has played an important role in providing the basis for the implementations undertaken in this paper. Firstly, it confirmed certain business needs commonly found among manufacturing enterprises: namely, the ability to remain agile and competitive in an ever-changing market. These are lofty goals however, and difficult to examine and judge in detail. More specifically, the study revealed Acme's particular requirements which were identified as the need for an IT support system that:

- (1) could handle the rapid addition of new sales agents
- (2) could handle the addition of new stock control centres
- (3) would allow changes to the business logic of the SOP to be made easily

These requirements are more specific, and will form the basis of the objectives for the implementation.

5.1 A Model For Sales Order Processing

The core processes identified at Acme that are involved in the SOP were first introduced in Figure 2. This model addresses all of Acme's core business processes. The implementation described in this work examined particular aspects of the overall model, concentrating on the interaction of a sales agent dealing with order requests and the stock control centres. The modified model is shown in Figure 6. Production Control was excluded since this is an entire field of research in its own right and was deemed external to the objectives of this

research. In addition, the greyed out areas in Figure 6, Dispatch and Manufacturing, represent processes that were not considered in our study, but would make excellent candidates for investigation and expansion in any future work.

Figure 7 depicts the implemented agent model for Acme's SOP. The fundamental operation is as follows: Following an enquiry from a customer to a SalesAgent, an OrderAgent is dispatched to the StockControlAgent, which is resident at a distribution point, where it requests the fulfilment of its order by passing over an Order object. The StockControlAgent queries the stock database to see if enough products are in stock. If there are enough products, the StockControlAgent then returns a DeliveryDate object to the OrderAgent which itself returns to its parent SalesAgent. The SalesAgent is then able to notify the customer of the delivery date.

If there are not enough products in stock to satisfy the order, the OrderAgent migrates to the manufacturing plant where it uses the Product ID encapsulated in the Order object and queries the BOM database for a list of sub-parts or raw materials required. This is then encapsulated within a WorksOrderAgent (again mobile) and dispatched to manufacturing while the OrderAgent returns with a DeliveryDate object containing a standard delivery date. If there are not enough raw materials in stock, agents within the manufacturing plant server generate a PurchaseOrderAgent that encapsulates details of all the required materials.

Our mobile object model is similar to that described above, the key difference being that the results from stock database queries are gathered from remote StockControlAgents by a mobile OrderObject guided by a specific itinerary. Instead of processing this information locally to the data source, it is returned to the SalesAgent for processing. The mobile object does not make autonomous decisions based on the acquired information.

6 The Systems

The experimental work described in this paper was implemented using IBM's Aglet Software Development Kit, a mobile agent development framework (Lange and Oshima 1998). The model, scenario and investigations described have been undertaken using both mobile agent and mobile object systems. The two systems consist of a selection of individual agents and objects. The types identified for the SOP model are SalesAgents, StockControlAgents, ManufacturingAgents, PurchasingAgents and DispatchAgents. There

are also two analogous but different components in each system. In the mobile agent system there are OrderAgents, whilst in the mobile object system there are OrderObjects. These are discussed in the following sections.

6.1 Sales Agents

Sales Agents are static, Graphical User Interface (GUI) based agents that are responsible for generating OrderAgents or OrderObjects and dispatching them to distributed agent hosts around the world to interact with StockControlAgents. These can be resident as a very slim client for sales persons working on terminals or NetPCs, or they can be hosted on a laptop for travelling sales persons. They are capable of keeping track of current orders that have been placed. In the mobile agent version the only logic contained within these agents is that required to create a new OrderAgent, with its accompanying Order. They are capable of maintaining a list of spawned OrderAgents, and thus which Orders have been fulfilled, or not. In the mobile object version they also contain the business logic required to process the results returned by their slave OrderObjects.

6.2 Order Agents

The OrderAgents represent the mobile elements in this system. As classified in Section 4.1, they are goal oriented, communicative and mobile. Each OrderAgent encapsulates a single order; and they are responsible for completion of that order. After creation they migrate to a new host to interact with a StockControlAgent. If the stock levels are unable to satisfy the order, they are able to migrate to a new host, and use the encapsulated Product ID to derive the Bill of Materials (BOM). The agent then migrates to the ManufacturingAgent's host. In future the agent would then invoke this manufacturing process or at least establish the required time for manufacture. Currently, this communication consists of a simple message and acknowledgement from the ManufacturingAgent. The valid outcome for the goal of the OrderAgent is reporting a delivery date for the order to the SalesAgent. In the future this may also include reporting a future time for delivery or an allocation for materials and an internal works order number and time to manufacture. OrderAgents require no interaction with a user and so have no GUI. Although quite simple, OrderAgent's make up the majority of the agent population in the system when it is executing dependant on the number of enquiries received by the SalesAgents. Potentially, there could be large numbers of mobile OrderAgents migrating through the network, attempting to fulfil their own

particular order.

6.3 Order Objects

OrderObjects initially appear to perform the same function in the mobile object system, as the OrderAgents described above. However, in contrast to the mobile agent system, it is more appropriate to view the mobile objects as mobile messengers. They are still able to migrate to a data source and take advantage of local interaction and all the advantages this brings, but they do not contain the business logic to autonomously process any results. They collect the stock level information and return to their origin to report findings to their parent SalesAgent, after which they are terminated.

6.4 StockControlAgents and MaterialsStockAgents

The StockControlAgents are another example of static agents, with no GUI. They are responsible for handling all requests for products, parts, or materials, and are interfaced to the stock control databases. As such they act as a wrapper to the data source, a communications bridge between the data and the agents system. All requests for stock allocation must be made through the StockControlAgents.

When designing StockControl Agents that could unify the variety of database systems which could be expected within a manufacturing enterprise, it became apparent that some of the required features of these agents were particular to each database, whilst others were generic to all StockControl agents. In considering this problem the use of a common 'Database Query Agent' was conceived which could be used as a base pattern for all StockControl agents in the system. The advantage of such a technique is the consistency and reusability inherent in using a pattern from which to build agents that are more complex. The Database Query Agent is discussed in (Papaioannou and Edwards 1999).

This architecture has been put to good use in the MaterialsStockAgents. They perform almost exactly the same role as the finished StockControlAgents, but are tasked with controlling the allocation of raw materials, and out-sourced parts. They are also connected to a Bill of Materials database, which the mobile OrderAgents and OrderObjects are able to query when having to request the manufacture of stock to fulfil an order.

6.5 ManufacturingAgents, PurchasingAgents, DispatchAgents

Currently these three types of agent are represented in the systems by static agents that are

communicative. They are able to simply acknowledge communication from other agents, and represent a definite avenue for further investigation and research.

7 Evaluation

The evaluation of this work was conducted according to Basili's GQM (Basili et. al. 1994, Solingen and Berghout 1999) method. This section includes an overview of the principle goals, questions and metrics identified before presenting and analysing the results.

7.1 The goal

“To evaluate the implemented system from the industrialist perspective, with respect to satisfying the industrial motivation to support system agility” (see Section 5)

Having stated the goal a workshop within the R.E.D. group at Loughborough University was held to generate a broad set of questions aimed at providing full coverage of the issues in order to derive a set of appropriate metrics by which the work could be evaluated. To fulfil the three goals specified in Section 5, the initial questions focused on system complexity and system flexibility. Using the Basilli method it is customary to develop a hierarchical set of questions which focus down to a set of questions that can be answered through tangible measures on the code.

Table 1 lists the focused questions generated, that were extracted from a large and varied set of questions generated by the workshop. Table 2 shows the set of relevant metrics that were generated from the workshop. The following sections describe how these metrics can be combined to provide two quantifiable measures of the degree of agility in the system. These measures can then be used to both analyse the object and agent systems and test the degree of agility of these systems in relation to the industrial objectives one, two and three specified in Section 5.0.

7.2 Conceptual Diffusion and Semantic Alignment

Conceptual Diffusion is a measure of: the degree to which a single concept or semantic abstraction in the application domain maps to many components in a software system. For example, metric (1) requires the identification of the information based concepts within the

real world, and a comparison with their counterparts in the software systems. If we use Order as an example, in both the agent and object system the concept of an Order is split over four separate components, scoring a conceptual diffusion rating of 4.

Semantic alignment between the real world abstractions and the components of a software system have been shown to be important when attempting to build flexible systems (Coutts and Edwards 1998). Conceptual diffusion is a measure of semantic alignment and can be used in a combination of metrics 1 to 4 to produce a compound measure of both these notions, where

$$SA = \left\{ \frac{I_s}{I_r}, \frac{P_s}{P_r}, \frac{M_s}{M_r}, \frac{S_s}{S_r} \right\}$$

SA is semantic alignment, I is information based abstractions, P is process based abstractions, M is mobile components, S is static components, s denotes in software and r denotes in the real world. Thus, $\frac{P_s}{P_r}$ is the ratio of process based abstractions in the software to the process based abstractions in the real world.

This metric can be used to analyse a system and to assess how well the software system reflects the semantics of the application domain, comparison with the ideal alignment of $\{1,1,1,1\}$ can be used as a measure to gauge how difficult it might be to understand the software, given an understanding of the application domain.

For the Mobile Objects System $SA = \{4,22/6,1/3,2/3\}$

For the Mobile Agent System $SA = \{4,21/6,1/3,2/3\}$

These results show that both the mobile code systems should be easy to understand, as the abstractions in the domain map reasonably well on to the components of the systems. The information abstractions from the real world are on average spread over 4 components in the implementation. When considering mobile and static component alignment, for both systems, a third of the components in the domain are modelled as mobile in the implementation, and two thirds of the static components in the domain are modelled as static elements in the implementations. The small difference in the two systems is shown when considering the semantic alignment of the processes. Here the mobile agent system is shown to have better semantic alignment than the mobile object system as the process logic for the Sales Order Process is contained within the mobile OrderAgent and not spread

across both the SalesAgent and the mobile OrderObject in the mobile object system.

7.3 Change Capability

In order to evaluate the agility of a system it is necessary to make changes to that system. The industrial requirements reported in section 5.0 specified agility in terms of adding new sales agents, new stock control agents and modifying logic associated with the sales order process. Metrics 8, 9 and 10 in Table 2 were used following modifications to the Agent and Object systems in line with these industrial requirements for agility.

An overall measure of Change Capability for each system is achieved through combining the results of the work in a single set, where

$$C_{\alpha \rightarrow \beta} = \left\{ \sum_a^b do, \sum_a^b ds, \sum_a^b di, \sum_a^b de \right\}$$

Change Capability CC, for a required change $\alpha - \beta$, is the set of the changes to the number of objects (o), the number of src files (s), the number of interactions (i) and the number of conceptual entities (ϵ), between states α and β .

Change capability can be used to compare systems or to get a measure of the changabilty of the system through its measured Change Capability relative to the ideal $\{0,0,0,0\}$. For the mobile object and mobile agent systems Change Capability for each requirement is as follows:

Handling the rapid addition of new sales agents:

Mobile Object and Mobile Agent system $CC = \{0,0,0,0\}$

Handling the addition of new stock control centres:

Mobile Object and Mobile Agent system $CC = \{3,3,1,2\}$

Allowing changes to the business logic of the Sales Order Process to be made easily:

Mobile Object $CC = \{2,2,0,2\}$

Mobile Agent $CC = \{1,1,0,2\}$

Again these results show that both systems are relatively easy to change. Adding new sales facilities requires only the instantiation of new SalesAgents which incurs zero changes to

the system code. New stock control centres can be added through a low number of changes that are the same for both systems. The difference between the systems becomes apparent when making changes to the Sales Order Process logic, which is contained in the single mobile OrderAgent in the mobile Agent system and in both the SalesAgent and the mobile OrderObject in the mobile object system.

8 CONCLUSIONS

This paper argues that mobile code can be an aid in the rapid integration and modification of the information systems that support distributed enterprises. This paper has characterised mobile agent systems as providing a mechanism to support mobile data *and* mobile logic. In contrast, traditional enterprise communication technologies are characterised as solely providing mobile data. Further, each of these is achieved through a different strategy. Mobile agents communicate through local interaction, whilst ORBs provide location transparency. From material and data collected during an industrial case study, a model was generated to support the manufacturing Sales/Order Process using both static and mobile agent and object technologies. A number of key agent types were identified and their role in the resultant system discussed.

The evaluation of the mobile agent system and the mobile object system described in the paper generated two new metrics which can be used to measure the Semantic Alignment of a system with its domain and the Change Capability of a system. Both were found to be useful when attempting to provide measures of a systems agility.

As would be expected, the results provide evidence that the mobile and static elements in the application domain are mapped well onto the mobile and static objects and agents in the implemented systems, and that generally both systems were very change capable. This has to be weighed against the fact that they were designed using OOP principles and built with flexibility in mind. There was no legacy to contend with. In addition, the authors support the idea that alignment of a system with its domain improves the understandability of a system.

Notwithstanding the considerations above, the work provides evidence that the mobile code paradigm is useful in providing a mechanism to better reflect the dynamics of a domain within the implementation of an IT system. Although there is little distinction between the mobile object and the mobile agent systems, the work identifies that the added

autonomy of a mobile agent system, useful for many established reasons highlighted in the paper, also supports agility better than mobile objects through a better Semantic Alignment and Change Capability.

9 ACKNOWLEDGEMENTS

The authors thank the EPSRC for their continued support, and the R.E.D. group members for their input to the metrics workshop.

10 REFERENCES

- Basili, V.R., Caldiera, G., Rombach, H.D., (1994), "The Goal Question Metric Approach", Encyclopedia of Software Engineering, pp 528-532, Wiley and Sons.
- Chess, D., Harrison, C., Kershenbaum, A., (1997), "Mobile Agents: Are they a Good Idea?" in Vitek (1997).
- Clements, P.E., Papaioannou, T. and Edwards, J.M., (1997), "Aglets: Enabling the Virtual Enterprise", Proceedings of the 1st International Conference on Managing Enterprises - Stakeholders, Engineering, Logistics and Achievement, ME-SELA '97, Wright, Rudolph, Hanna, Gillingwater and Burns (eds), Mechanical Engineering Publications, Loughborough University, July 1997, pp 425-432, ISBN 1-86058-066-1.
- Coutts, I., Edwards, J., (1998), "Support for Component Based Systems: Can Contemporary Technology Cope?", Intelligent Systems For Manufacturing, Edited by L.M. Camarinha-Matos et. Al., Kluwer Academic Publishers, ISBN 0-412-84670-5, pp279-288. "Basys paper"
- Franklin, S and Graesser, A., (1996) "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proc. of the 3rd Int. Workshop on Agent Theories, Architectures, and Languages, Berlin, Springer-Verlag.
- Gray, R., (1997), "Agent Tcl: A flexible and secure mobile-agent system", PhD thesis, Dept. of Comp Sci, Dartmouth College.
- ISO, (1992), International Standards Organisation, "Basic Reference Model of Open Distributed Processing, Part 1: Overview and guide to use", ISO/IEC JTC1/SC212/WG7 CD 10746-1.
- Kiniry, J., Zimmerman, D., (1997) "A hands-on look at Java Mboile Agents", IEEE Internet Computing, Vol (1) 4, July/August, 1997.
- Lange, D.B., Oshima, M., (1998) "Mobile Agents with Java: The Aglet API", World Wide Web Journal.
- Lange, D.B., Oshima, M., (1999) "Seven Good Reasons for Mobile Agents", Comm. ACM, Vol 42 (3), p88-89.
- Minar, N., Gray, M., Roup, O., Krikorian, R., Maes, P., (1999), "Hive: Distributed Agents for Networking Things", Proceedings of. ASA/MA '99.
- MML, The Mobility Mailing List, <http://mobility.lboro.ac.uk>
- ObjectSpace: Voyager Core Package Technical Overview (1997), <http://www.objectspace.com/Voyager>
- OMG (1994), Object Management Group, "The Common Object Request Broker: Architecture and Specification", Object Management Group Inc. 492 Old Connecticut Path, Framingham, MA., USA 1994.
- Orfali, R., Harkey, D., Edwards, J., (1996), "The Essential Distributed Objects Survival Guide", John Wiley and Sons Inc. New York. USA 1996.
- Papaioannou, T., Edwards, J.M., (1999) "Using mobile agents to improve the alignment between manufacturing and its IT support systems", Journal of Robotics and Autonomous Systems, Vol 27, pp45-57.
- Papaioannou, T., Minar, N., (1999), "Mobile Agents in the Context of Competition and Cooperation", Proc. MAC3 workshop, part of Autonomous Agents '99 conference, Seattle.
- Rothermel, K., Hohl, F., (eds) (1998), "Mobile Agents", Proc. 2nd International Workshop, Stuttgart, Germany, September 1998, Lecture Notes in Computer Science 1477, Springer-Verlag.
- Solingen, R.V., Berghout, E., (1999), "The Goal/Question/Metric Method", McGraw Hill, ISBN 0-07-709553-7
- System Software Associates Inc., (1995), "BPCS Client/Server Distributed Object Computing Architecture".
- Straßer, M., Baumann, J., Hohl, F., (1996), "Mole - A java Based Mobile Agent System", in Proc. ECOOP'96 workhop on Mobile Object Systems.
- Sun Microsystems Inc., (1998), "Java Remote Method Invocation Specification", Revision 1.50.
- Vitek, J., Tschudin, C., eds, (1997), "Mobile Object Systems, Towards the Programmable Internet", Lecture Notes in Computer Science 1222, Springer-Verlag.
- White, J. E, (1994), "Telescript technology: the foundation for the electronic marketplace", White Paper, General Magic Inc., 1994, also appears in "Software Agents", J. M. Bradshaw (Editor). MIT Press, 1997. ISBN 0-262-52234-9.
- Waldo, J., Wyant, G., Wollrath, A., Kendall, S., (1994), "A note on distributed computing", Sun Microsystems Technical Report SML 94-29, 1994.

11 Tables

TABLE 1. Questions generated using the Basilli GQM Method

Generated Questions	Metric Number
How well does the system support change?	
How easy is it to understand the system?	
How many business entities map onto data abstractions	(1)
How many business processes map to software methods	(2)
Which real world entities that are mobile are also mobile in the system	(3)
Which real world entities that are static are also static in the system	(4)
How many components are there in the system	(5)
How many lines of code are there	(6)
How many comments are there	(7)
How easy it was to modify the system?	
How many conceptual entities must be changed - for example requirement a)	(8)
How many objects must be changed	(9)
How many src files must be changed	(10)
How many interactions must be changed	(11)
How many components are there in the system relative to the size	(5) + (6)
How many real world entities map to a software component	(1)+(2)+(3)+(4)
How many components must be changed	(9)
How many interactions must be changed	(11)
How many inter-entity connections are there	(12)
How many methods of the object are public	(13)

TABLE 2. Metrics Generated using the GQM Method

Metric	Nature of Metric
(1)	Identify information-based abstractions in the real world. Compare with info based abstractions in the software
(2)	Identify process-based abstractions in the real world. Compare with processes evident in the software.
(3)	Identify mobile elements of the real world, compare with mobile elements in the software
(4)	Identify static elements of the real world, compare with static elements in the software
(5)	Count the components
(6)	Count lines of code
(7)	Count comments, and get ratio of comments/method
(8)	Count changes for each requirement
(9)	Count changes for each requirement
(10)	Count changes for each requirement
(11)	Count how many files are changed for each requirement
(12)	Count number of inter object method invocations
(13)	Count number of public methods

12 List of Figures

Figure 1. Acme's business processes on receipt of an Order

Figure 2. Top level view of business processes within Acme

Figure 3. Communication between components via an ORB

Figure 4. Comparison of distribution mechanisms

Figure 5. Example of a mobile agent migration, to be local to a data source

Figure 6. Modified Process Model for Acme

Figure 7. Agent Sales Order Process Model - with example routes for OrderAgents

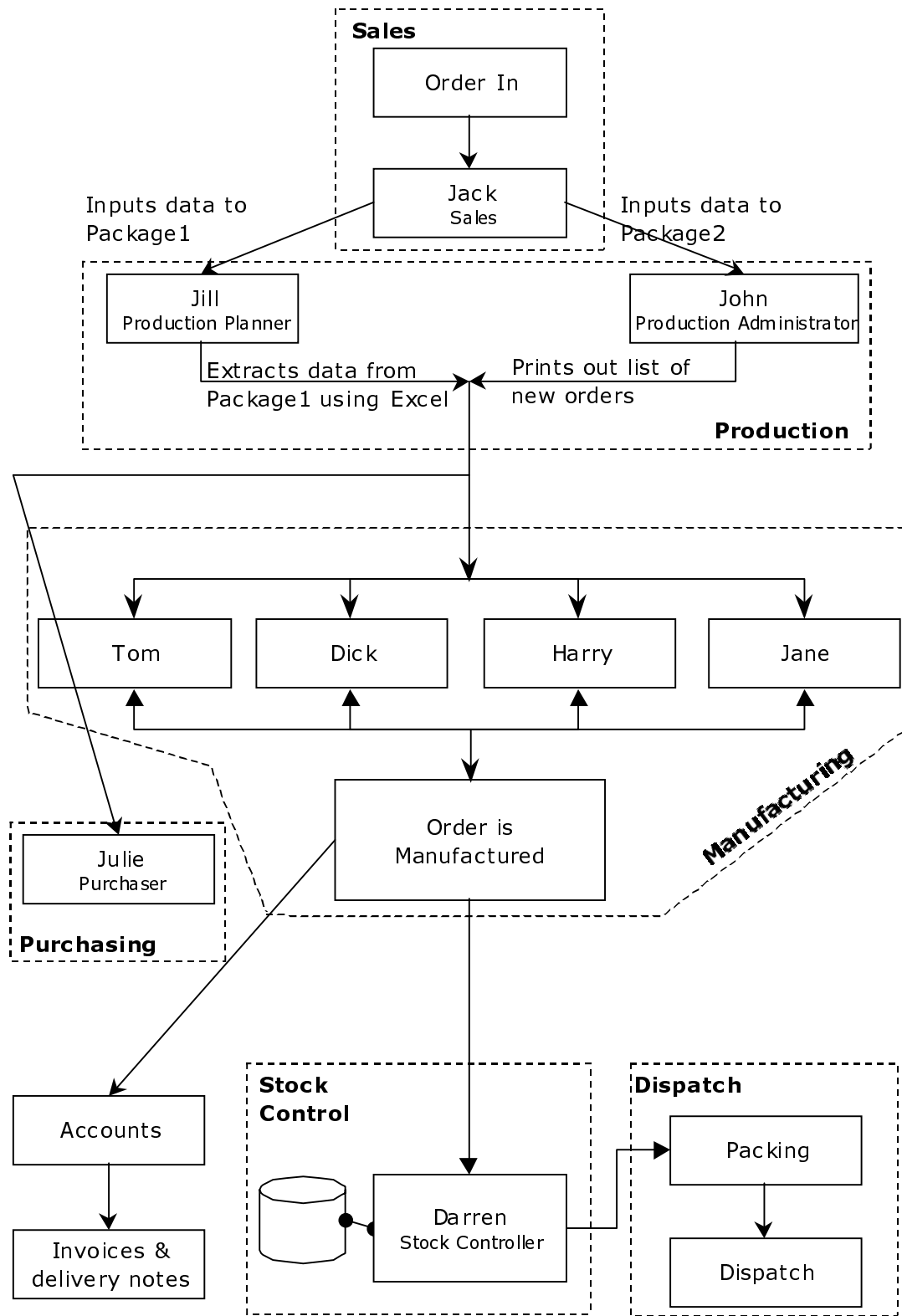
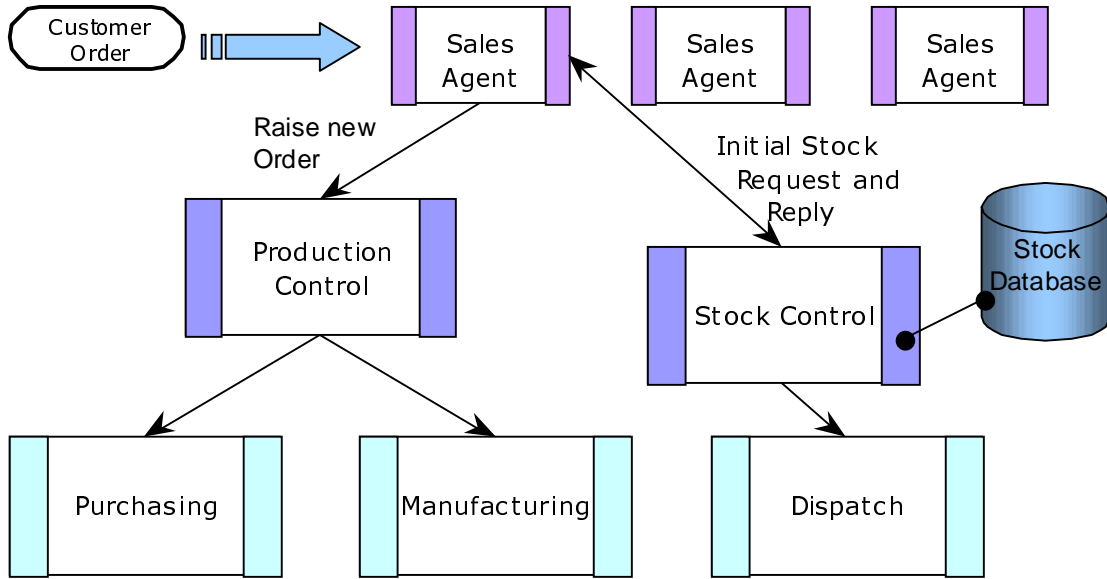


Fig 1. Acme's business processes on receipt of an Order



Example Walk through

A new Customer Order is placed with a Sales Agent. The Sales Agent then interrogates Stock Control to see if the order can be fulfilled from the existing stock. If it can, a new Order is raised and the items are allocated to that order number before being dispatched to the customer, along with an invoice.

If the items are not in stock, then the order is passed to production control where again, an Order is raised. Accompanying this Order is a new Works Order for the required manufacturing of the requested products, or product parts. The Works Order is then passed to manufacturing for completion, and if necessary purchasing for replacement of raw materials. Once the product or parts are completed, they are booked into Stock Control before being checked out again for dispatch. The standard delivery time at Acme is three weeks, unless the order is being specially manufactured to specifications submitted by the customer.

Fig 2. Top level view of business processes within Acme

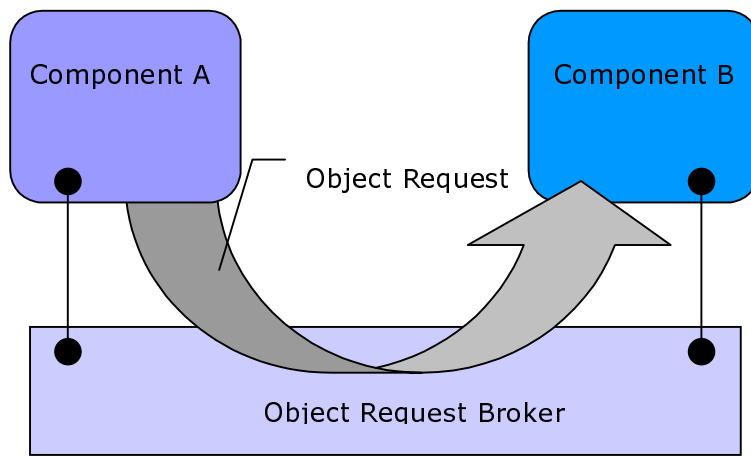


Fig 3. Communication between components via an ORB

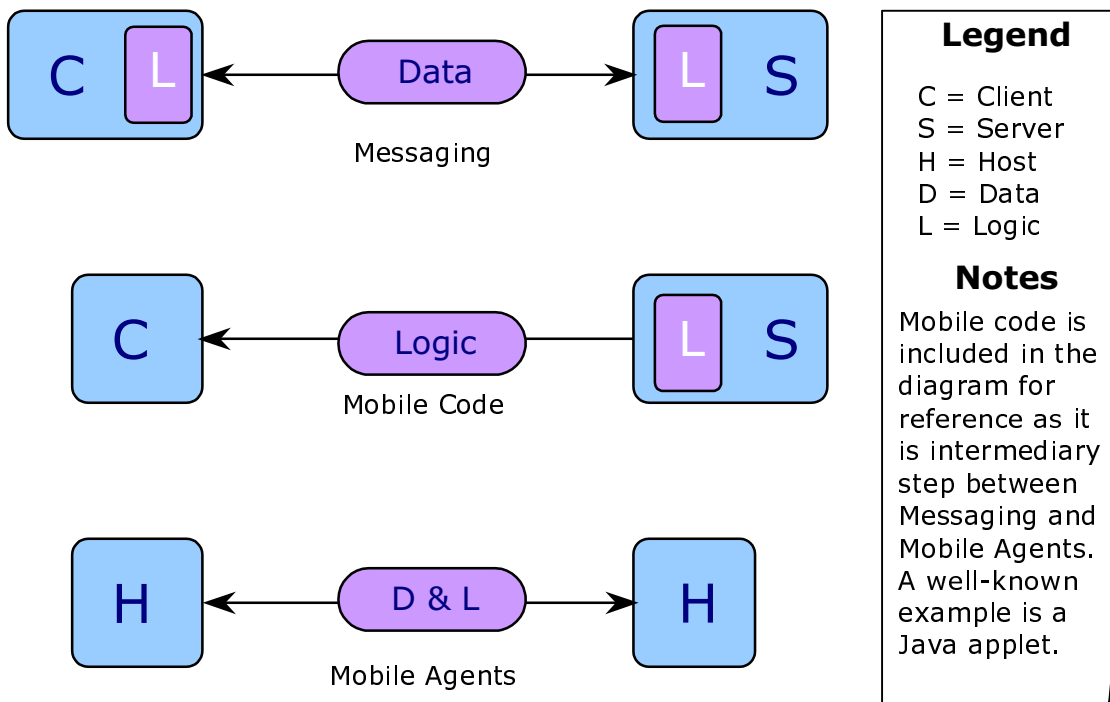


Fig 4. Comparison of distribution mechanisms

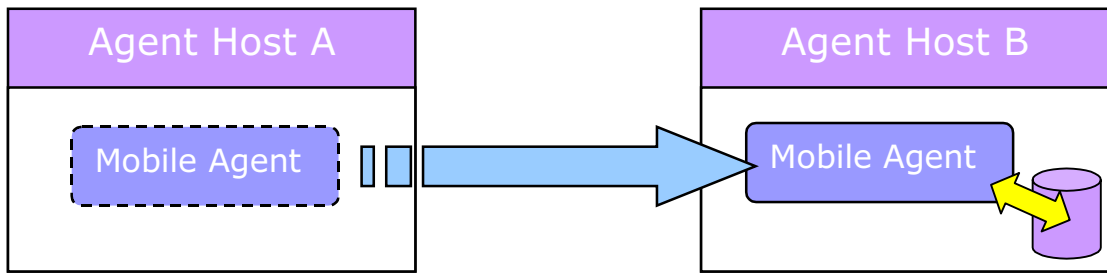


Fig 5. Example of mobile agent migration, to be local to a data source

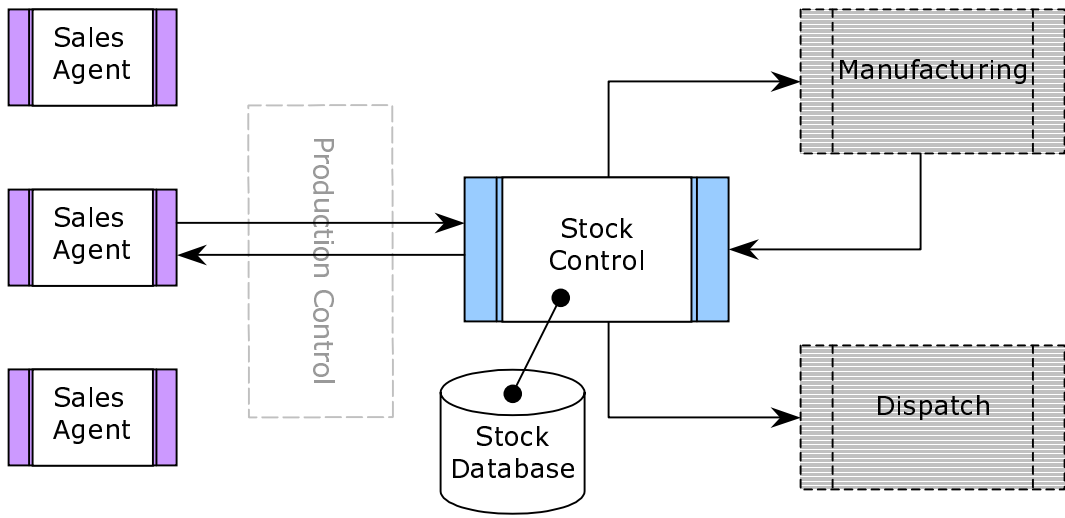


Fig 6. Modified Process Model for Acme

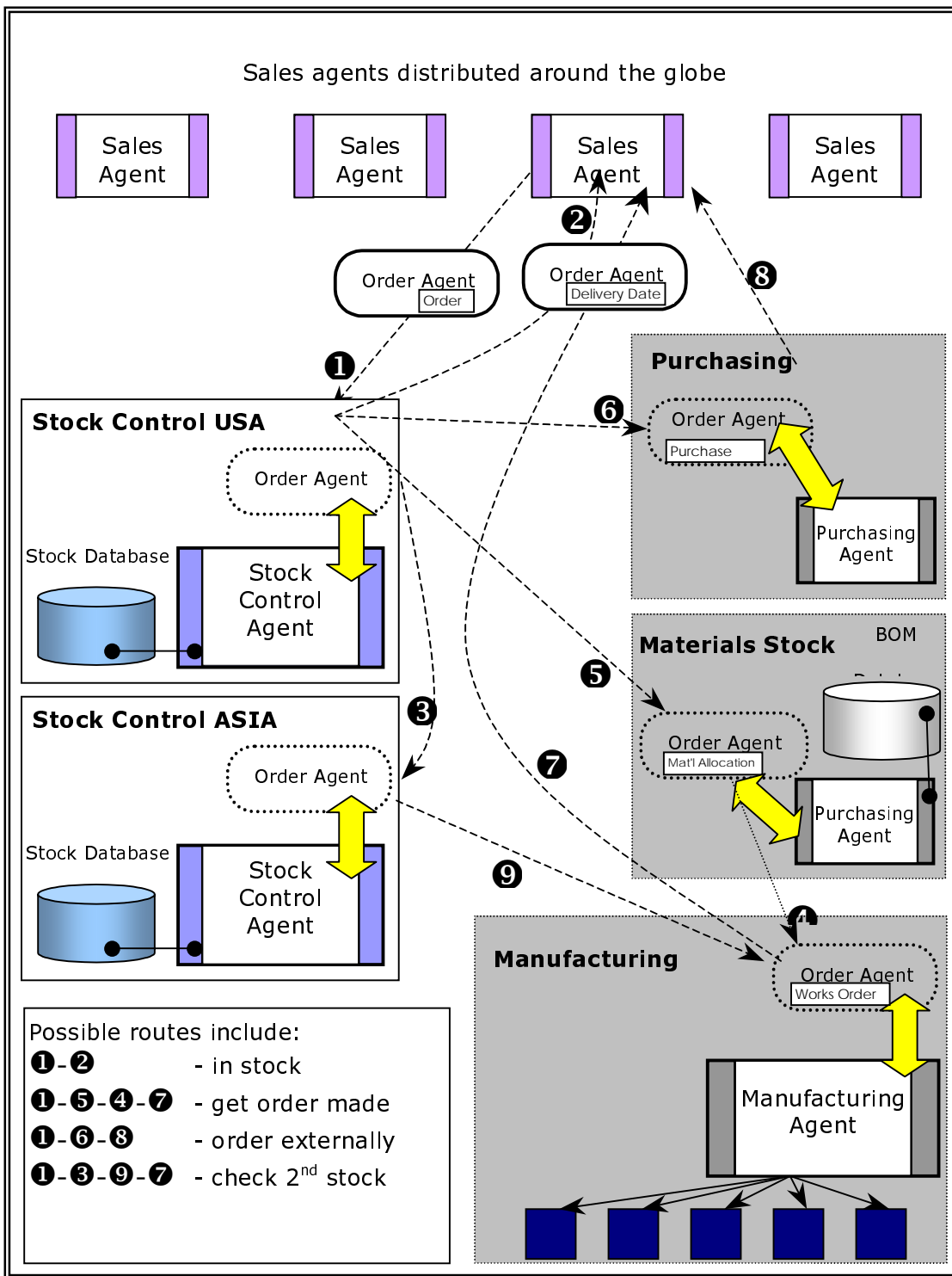


Fig 7. Agent Sales Order Process Model – with example routes for OrderAgents