Int J Adv Manuf Technol (2007) 33: 354–364
DOI 10.1007/s00170-006-0462-1

ORIGINAL ARTICLE

Chi-Yu Huang · Alan Holt · John Monk · Kai Cheng

The application of dependency management in an integrated manufacturing network framework

Received: 20 July 2005 / Accepted: 15 December 2005 / Published online: 23 May 2006
© Springer-Verlag London Limited 2006

Abstract Research into the effectiveness and performance of the mobile agent (MA) technology as a means of managing a domain of manufacturing devices has been conducted. However, it has been shown that MAs are not without their scalability issues. A strategic agent travelling algorithm will bring performance improvement as it allows agents to identify the best migration path in order to minimise the total expected time of searching for the desired information. The applications of process-driven dependency management along with MA techology are examined as methods for optimising the efficiency of retrieving critical information in the manufacturing environment. An integrated framework is developed to investigate the alignment of network management paradigm to the strategic management decision.

Keywords Dependency management · Mobile agent itinerary · Process-driven

C.-Y. Huang (✉)
DL Consulting,
Hamilton, New Zealand
e-mail: chi@dlconsulting.co.nz

A. Holt
Department of Computer Science and Mathematics
University of Waikato,
Hamilton, New Zealand
e-mail: aholt@cs.waikato.ac.nz

J. Monk
Faculty of Technology, The Open University,
Milton Keynes, UK
e-mail: jmonk@open.ac.uk

K. Cheng
Faculty of Information and Engineering Systems
Leeds Metropolitan University,
Calverley Street,
Leeds, UK
e-mail: K.Cheng@leedsmet.ac.uk

1 Introduction

The development of information technology (IT) leads to an increase in use of heterogeneous networking products in manufacturing enterprise and business environments. In our previous research, a strategic framework based on the mobile agent (MA) technology at the network management level was proposed [1]. The major purpose of the research was to examine how the decision makers and managers can effectively schedule and optimise the enterprise's strategies through the monitoring of network activities. Networks, along with the appropriate software, give manufacturing devices the ability to communicate, which in turn enables them to be managed from a remote location. However, in a distributed system, networks introduce delays (transmission delays, propagation delays, etc.) that can directly affect any management system's ability to exercise effective control of a manufacturing environment.

The MA technology was examined in [1] as a means of improving the efficiency and performance over that of traditional network management paradigms, such as simple network management protocol (SNMP) and manufacturing automation protocol (MAP). MAs are expected to have the intelligence to decide when to move, as well as the underlying infrastructure to support and execute any request. By moving to the location of an information resource, the MA can search the resource locally, which eliminates the need for the transfer of intermediate results across the network and reduces end-to-end latency. Nevertheless, it is still challenging to determine the itinerary strategy at the time the agent is instantiated, especially in a dynamic environment. Hence, what is required is a strategic travelling algorithm that allows an agent or a small group of cooperating agents to identify the best migration path in order to minimise the total expected time of searching for the desired information.

This research will investigate whether or not the dependency relationships between the manufacturing process and its associated manufacturing resources have the potential to bring any productivity benefits. The discussions will focus on behavioural comparisons of the mobile agent paradigm in practical use. The performance parameters considered are the data generated in the network traffic in a domain between the management station (or platform) and managed devices as well as the overall delay in retrieving useful operational variables as they are significantly affected by their behaviour.

Manufacturing network systems can be described as "safety-critical" systems. Implementing a change in such systems can be complex and difficult, and so experimentation with a real system can be both costly and risky. Therefore, analytical models are used in this paper to examine and evaluate the proposed framework.

The structure of this paper is organised as follows. Section 2 will discuss the travelling behaviour of an MA and its challenges. Section 3 investigates the concepts of dependency management and discusses its role in this research. In Section 4, a scaleable platform-based network architecture designed to cope with the dynamics of the manufacturing environment is presented in detail. A number of analyses regarding the dependency applications in the proposed framework are conducted and demonstrated in Section 5.

2 Travelling behaviour of the mobile agent

2.1 Resesarch in agent travelling algorithms

An MA is not bound to the system from which it is initiated. It is often required to travel between multiple managed nodes or computer nodes in order to perform its tasks. MAs are expected to have the intelligence to decide when to move, as well as the underlying infrastructure to support and execute any request. Research on agent travelling algorithms includes the highest probability first search (HPFS) algorithm [2] and the travelling agent problem (TAP) [3, 4].

Chen [2] applies the HPFS algorithm in a mobile computer environment. The algorithm is used to estimate the time spent on locating the target agent. The probability is a value which shows the possible chance that the agent might stay. According to their research results, Chen found out that a strategy of querying to the server with the highest probability among the servers will consequently consume less search time and network overhead than blind search strategies.

Moisumi [4] and Brewington [3] propose a travelling agent problem (TAP) which is based on the analogy of the travelling salesman problem (TSP) [5]. The algorithm is mainly used to decide the sequence of machines to visit in order to minimise the total expected time of searching for the desired information. A known probability representing the possible chance of being able to complete the agent's task successfully was introduced. Once the required information is found, the agent will finish its journey instantaneously and return to the site where it was initiated. All these research works focus on finding a managed object (or objects) on a particular managed device in a domain of devices.

2.2 Research synopsis

In this paper, the research is concerned with monitoring the state of a manufacturing system which is reflected by the managed objects (manufacturing devices or recourses), dynamically changing with time, of every device in the system's domain. Each of the devices will vary in importance regarding their contribution to supporting the system. Accordingly some devices will have higher dependency than others and will play a more crucial role in maintaining the overall manufacturing system. The problem to be addressed in this paper, therefore, is that of sampling managed objects at a sufficient rate in order to capture data such that it accurately reflects the current state of the system. An application-oriented strategy applied to this scenario will be discussed in this research.

3 Dependency management

Dependency management involves specifying the relations between processes as well as those related manufacturing resources which are used to implement those operations. In this research, the term "dependencies" represents the interrelationship of resources or information between activities in either a physical or a logical way. As the information is normally dynamic and can change over the lifetime of the dependency, a dependency management framework has to offer flexibility to change dependency bindings in real-time. The information may come from the best knowledge of the process designers and engineers or heuristic experience, but this can not present the dynamic nature of the dependency. Much information is available in the literature describing the use of dependency models in various areas [6–8].

In this paper, we will demonstrate a model to retrieve dependency information from the manufacturing system in a real-time and dynamic manner.

3.1 Applications of dependency management

In the manufacturing environment, a manufacturing system includes a set of machines or resources performing different operations in order to implement manufacturing processes. In the field of service application management, an important requirement is the ability to perform effective fault and performance management within a networked environment. It is the capacity to understand how faults occurring in the resources of one layer affect the working of resources in another layer. Several research works have utilised the concept of dependency management in order to state the capabilities of applications service management and reveal its problem issues [7–11]. It is commonly believed that once a dependency between related components exists, the root cause may be tracked down to a resource on which a certain service is dependent. Hence, It is expected to detect system problems, isolate their root causes and identify proper repair procedures.

Analysis of the dependency of business and manufacturing systems can be found in [6, 12]. For example, in Jin [6], an agent-based framework - ActivePROCESS - is proposed to capture work dependencies between activities at the engineering level. This includes the clear definition of process activities, their interrelationships, responsibilities and performance requirements. Kim [12] discusses how the managing dependencies among interdependent elements within an organisation can help to improve business-process management.

So far, little research has been done into analysing the impact of system-level resources, while changes to the production plan or customer requirements occur at the enterprise-level, especially those elements which cross systems and domain boundaries.

In this research, we investigate the dependencies between manufacturing processes and resources, and explore how their inter-dependencies may affect the overall enterprise strategy.

3.2 Dependency management in the enterprise manufacturing environment

Typically, the required information or data may be located at a single domain or may be spread across different domains. Applied to the business and manufacturing environment, it is clear that these relationships can be defined as:

- Intra-organisation dependencies: These are dependencies occurring among functional business processes within the same organisation. For example, sales activities normally associate with inventory, production and manufacturing planning activities
- Inter-organisation dependencies: These dependencies are between functional processes across boundaries between partner companies. Such as the activities occurring within a supply chain network and outsourcing companies

In the manufacturing environment, a system consists of a multitude of functions which are installed on different computers and interconnected by a complex computer communication technology. This system supports the organizational planning and scheduling, manufacturing processes control and monitoring for producing goods and services, people for performing tasks, strategic and technological planning. Here, the information flow has to be controlled in a precise manner in order to provide the customer with a high quality product and good customer service. These entities are highly related and interdependent on each other. The major focus of this research is the analysis of the processes view.

For a certain business or manufacturing plan, several designed processes are required. The definition of dependencies is used to identify the relations between processes as well as the related work resources used to implement these processes. In order to accomplish a manufacturing process, various activities are normally needed. If a process requires a particular activity, it is said to be dependent on that activity. If more than one process depends on the same activity, each dependent component has its own dependency.

Transferring the above scenario to the level of network management, those work components (for example, tools or resources) can be regarded as networked elements. For a manager to achieve effective and efficient coordination with network monitoring, it is helpful if the dependencies between work components and processes are well understood. A proposed network management platform particularly applying to the manufacturing environment will be discussed in detail Section 4.

3.3 The challenges of dependency management

Dependencies have a certain set of properties. One of the most important of these is that they are dynamic and have the ability to change over the lifetime of the dependency. To perform dependency management, the dependencies must be accessible to management applications. For analysing or demonstrating process dependencies, the available processes and their dependencies must be visible from the outside. During execution, the dependencies of the process can be variable. Relevant components may become unavailable, migrate or be upgraded. A process must have the capability to change its own bindings in order to respond to exceptional conditions. A dependency management framework therefore has to offer flexibility to change dependency bindings at execution time.

3.4 Mechanisms for dependency analysis

Since it appears difficult to identify the interactions between the various functions and processes within a manufacturing system, there are numerous techniques for modelling manufacturing processes and the management of their dependencies, which are described as below:

- Virtual clustering mechanism: This method groups similar records together under the same category. Applying the clustering approach to the distributed manufacturing environment, manufacturing processes and subtasks can be clustered into various domains, in which correlated functions will probably be clustered together. Thus, this method can identify different domains for an abstract high level overview of the whole manufacturing enterprise or business. Virtual clustering is a dynamic, emergent function-driven group behaviour of the multi-agent system in which agents can dynamically participate in a specific problem-solving cluster (or clusters). In general, there is prior information available about the data (for example, statistical model, historical information, etc.) so that the decision maker can make a few assumptions about the data. The variety of techniques for representing data, measuring proximity (similarity) between data elements, and grouping data elements into clusters has attracted a lot of attention in the research field [13–15].

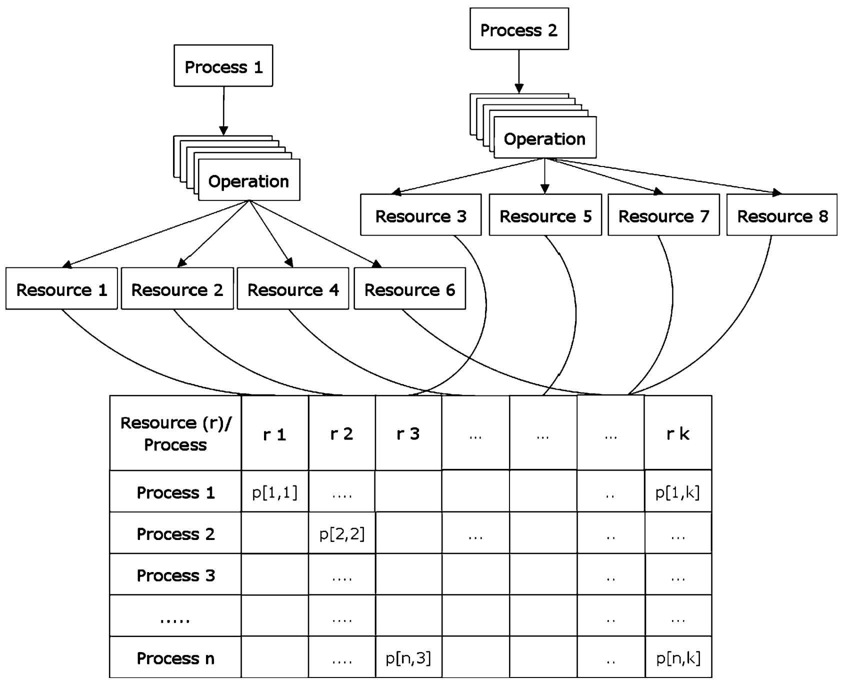
– Dependency graph: The concept of dependency graphs is based on the theory of graphs. The basic premise underlying the dependency graph is to model a system as a directed acyclic graph, in which nodes represent system components or resources (for example, services, applications, software, hardware and networks) and weighted directed edges between nodes represent their dependencies. The strength of dependencies can be denoted as strong (S), medium (M) and weak (W) against the edge between two nodes. Bagchi [9] and Kar [16] introduce this method to show how an application service is dependent on lower layer services/resources in a networked manufacturing environment. This graphical representation is based on static dependency analysis techniques. Dependency graphs provide a straightforward way of identifying possible root causes of an observed problem - one must simply trace the dependency edges from the problematic node (or entity) to discover all of the potential root causes. The details of a dependency graph can be constructed and stored in a dependency database. By querying the dependency database, an application agent is able to construct a list of all the resources on which a certain manufacturing process depends. The end result is that each working process has associated with it a list of resources in the physical layer that provides the basis for implementing that process service.

– Dependency structure matrix: The dependency structure matrix (DSM) is also known as the design structure matrix, which was introduced by Steward in 1981 and has been applied to several studies [17–19]. In this paper, we will use the acronym DSM to refer to dependency structure matrix. DSM represents and visualizes relations and dependencies between tasks and activities, components and subsystems, and among people and teams [19]. By reading across a row of the matrix, various interactions can be observed through the cell contents corresponding to each cross-referenced column. It provides an overview of the different kinds of relationship, depending on what variables are considered. Danilovic and Borjesson [20] recognises that the matrix representation makes it possible to create a complete model of information flow and dependency analysis in describing and analyzing complex projects. It allows tasks to be either coupled or independent. The traditional DSM methodology is based on using square-designed matrices. However, some researchers evidently use a rectangular matrix layout [17, 20]. Each cell can be given a marked or numerical value, termed points of interaction (POI), indicating a task dependency or their importance. These numerical values can represent the degree of dependency between two variables or elements.

3.5 Discussions of dependency models

A number of mechanisms used in capturing dependency details are examined above. However, the method of virtual clustering is complicated due to differences in assumptions and contexts between different user groups. Furthermore, with this method, it is not easy to express the degree of dependencies between correlated elements.

Similarly, the method of dependency graph can not explicitly express the degree of correlated components. It only provides a static view of the overall system, and in an



Resource (r)/ Process	r 1	r 2	r 3	...	...	...	r k
Process 1	p[1,1]	....				..	p[1,k]
Process 2		p[2,2]		...		..	...
Process 3		....				..	...
.....		....				..	...
Process n		....	p[n,3]			..	p[n,k]

Fig. 1 An example of a process-resource dependency structure matrix

environment with a large number of counted processes and resources, the resulting dependency graph would be complex and inflexible. Therefore, the method of DSM was chosen at the pre-implementation stage of our analysis.

In this research, a rectangular DSM is used for analysing the inter-dependency of overall manufacturing processes, their associated operations and working resources. The numerical system is adopted in order to reveal the individual dependency of the operations on a certain process. This process-resource DSM provides an insight into resources that interact with each other and how those processes are affected. A simple example is demonstrated in Fig. 1.

Figure 1 shows a sample process-resource DSM, a 4×7 matrix. Theoretically, this matrix is built upon 28 POI. Each of these POIs can contain any type of relation or none at all. Manufacturing processes appear identically labelled in rows, whereas the designed resources arranged in the columns of the matrix. $p(i,j)$, shown in the cell, simply identifies and represents the perceived level of dependency, where $0 \leq p(i,j) \leq 1$. The value of $p(i,j)$ regarding the interrelationship between a manufacturing processes i and associated resource j can be obtained. For example, by the calculation of the required utilisation of the resource in order to execute a particular process (or processes). It can be retrieved from the historical log files or dynamically in real-time. The bigger the number, the higher the dependency represented. If some rows contain a lot of identified POIs, this obviously shows that this process may have a serious impact on the overall system under analysis. In some cells, multiple $p(i,j)$ could represent simultaneously perceived dependencies from different points of view. This means that different interpretations were defined by people who were asked to identify and estimate the impact of identified dependencies. This complex area, however, lies beyond our research scope.

In the above section, a number of approaches are discussed which can be applied to dependency management in a process-driven manufacturing environment. Although our research interest is not in evaluating the efficiency of these approaches, this paper will demonstrate the behaviour of retrieving dependency details in a dynamic manner and the use of these dependency details to facilitate company decision-making.

4 A scalable manufacturing enterprise agent platform

In our previous research, we proposed a scalable and flexible integrated framework to investigate how collaborative high-level decision-making procedures can be optimized by considering the real-time information retrieved at shop-floor level. This information may come from the managed components in the managed physical layer such as robots, machine drilling units, cell controllers, numerical controllers, and programmable logic controllers which are interconnected through computer networks. The proposed framework is shown in Fig. 2.

4.1 Dependency applications

In this proposed framework, a *mobile agent platform* (MaP) architecture is developed in the management platform layer. The MaP is a software package or platform for the development and management of mobile agents (MAs). In addition, the MaP is mainly concerned with the processing of management data and information collection, providing key management services like monitoring and

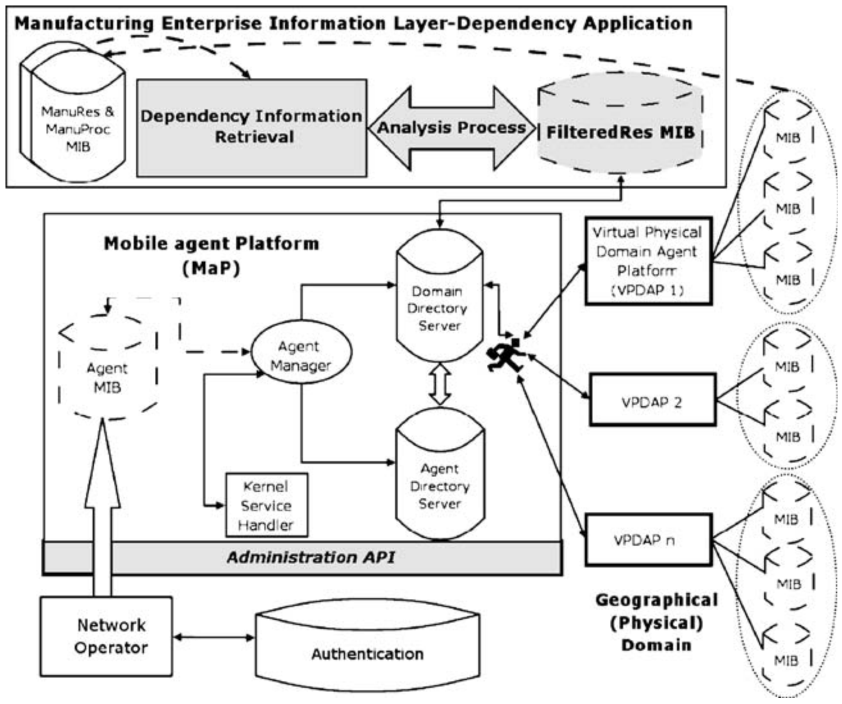


Fig. 2 The view of an integrated manufacturing management framework and agent platform

controlling. The below will describe the functionalities of the MaP in details. This paper will focus on the discussion of the dependency application in the proposed framework.

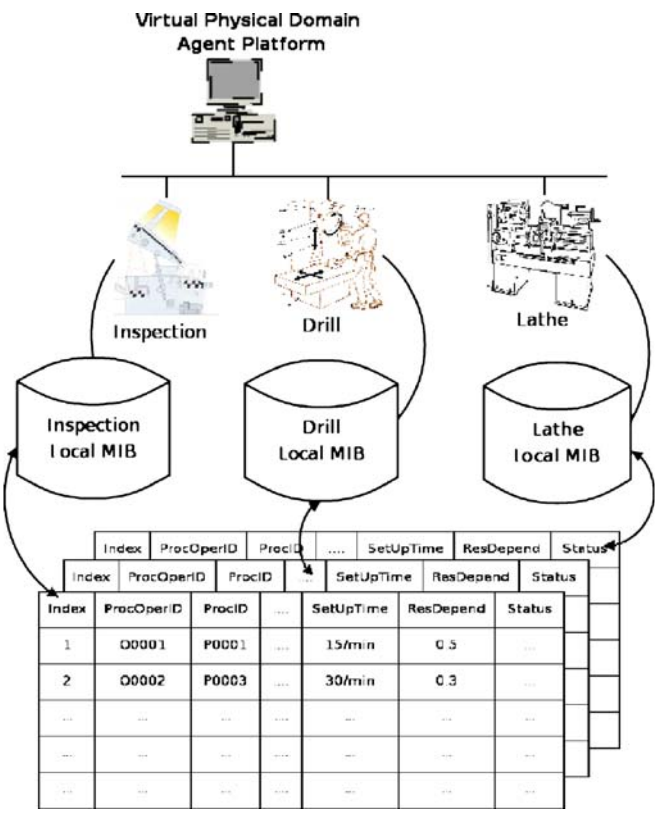
– Application dependency analysis. This dependency application mainly provides details of relevant resources accompanied by their degree of dependencies to the specific process. We assume if the dependency between a device (resource) and the process is high, it means that the process is heavily reliant on (tightly coupled to) the device. Otherwise, the associations (relationships) between the process and resources are loosely coupled. The respective dependency metric will be stored in the local management information base (MIB) and then be sent to the FilteredResMIB in the later stage. The FilteredResMIB is generated in the highest layer of the framework and then turn to a directory service at this layer.
– Agent manager (master). The manager acts as the master and can launch mobile agents (slaves); identify the travelling itinerary of MAs; monitor and control the network situation; and display their results.
– Mobile agents (slave agents). MAs are software program objects with a unique ID, capable of migrating between hosts where they execute as separate threads and perform their specific management tasks.
– Directory service. A directory is essentially a specialised, server-based database and provides a lookup function. In this framework, two directory services are defined. One is used to store the domain information, whereas another stores dispatched agents' details.
– Kernel service handler. The kernel services implement "housekeeping" operations and management services including log, alarm, security, state monitoring, and event/trap services. The details of these services are described in Stamatelopoulos [21].
– Virtual physical domain agent platform (VPDAP). At each physical domain, a VPDAP is designed for handling certain key management services such as agents' authentication and issues concerning the processing of the management data and information collection, monitoring and controlling etc. Furthermore, it will reply to the agent manager with detailed and summary information.
– Local MIB. It is an associated local MIB in the managed physical layer. The local MIB contains all the detailed information that concerns the respective resource. For example, it stores effective values of the resource parameters, resource status, resource capacity, usage and dependency values in the manufacturing system.

This is a brief outline of the MaP functionality which is part of our proposed framework. A more detailed description with respect to the holistic framework can be found in [1].

4.2 Dependency information

In the manufacturing system, each managed manufacturing resource can normally be used to support one or more manufacturing processes. Thus, there will be multiple dependencies for a certain manufacturing resource. In order to cope with this complexity, in the proposed framework, we specifically designed a ManuMIB [1] (manufacturing management information base) in which the dependency metric can be stored. It is based on the assumption that for each local MIB, there is a table recording the details about the process or processes executed on each individual manufacturing resource and its associated dependency. We suggest that the dependency metric is a definable property and will vary according to the manufacturing scenario. For example, the "utilisation" indicates the proportion of the overall capacity that has been used to execute a certain operation. A resource with a high utilisation will have a corresponding high dependency. This example explains a fairly simple relationship between a state variable (resource utilisation) and dependency. However, dependency metrics could be based upon much more complex formulae (possibly determined by the human manager/engineer).

The example in Fig. 3 shows a partial manufacturing process flow for the plug assembly [22]. It demonstrates



Index	ProcOperID	ProcID	....	SetUpTime	ResDepend	Status
1	O0001	P0001	....	15/min	0.5	...
2	O0002	P0003	....	30/min	0.3	...
...	...	...	....	...	...	...
...	...	...	....	...	...	...
...	...	...	....	...	...	...

Fig. 3 An example of manufacturing processes with associated resource-process local ManuMIB

the process flow from inspecting the material to the lathe processing stage. As can be seen from the figure, there is a local MIB for each resource which records the details about the associated processes, the resource's physical location, setup time, work hour, make time, status etc.

This example suggests that each manufacturing resource may be used to support one or more processes (operations). The value of dependency can then be calculated based upon the dynamic behaviour of the resource and its utilisation. The information from these local MIBs can be stored in a central ManuProc MIB. The representation of ManuProc table is shown in the MIB format below:

```
ManuProcTable OBJECT-TYPE
    SYNTAX SEQUENCE OF ManuProcEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The (conceptual) table of processes contained
        by implementing manufacturing projects"
ManuProcEntry OBJECT-TYPE
    SYNTAX          ManuProcEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry for a single process on the
        manufacturing system"
    INDEX {ManuProcIndex}
        ::={ManuProcTable 1}
ManuProcEntry::=
SEQUENCE {
    // Unique Process ID
    manuProcID              DisplayString,
    // Process Start Time format DDMMYYHHMM
     [SS]
    manuProcStartTime  UTCTime,
    manuProcOperation  SEQUENCE OF
    {
      // Unique Operation ID
      manuProcOperID        DisplayString,
      // The resource ID used to support
      // the operations
      manuProcResID         DisplayString,
      // The resource description.
      manuProcResDesc    DisplayString,
        ...
      // The physical location of this resource.
      manuProcResLocation IpAddress,
      // The current status for the resource.
      manuProcResStatus    Status,
        ....
      // The dependency for the process
      // dependent on that resource.
      manuProcResDependency REAL,
    }
    // Process finish time, format DDMMYYHHMM
     [SS].
    manuProcEndTime    UTCTime,
    // Process              status
    manuProcStatus          Status,
}
```

A more completed vision of this proposed framework can be seen in [1]. The status of an individual dependency can vary over its lifetime, therefore, retrieving dependency information must be dynamic and in real-time. In order to perform effective dependency management, dependencies must be accessible to management applications. Moreover, management applications can utilise these dependency details to monitor and control the manufacturing resource more effectively and efficiently.

5 Analyses of dependency applications in the proposed framework

In this analysis we consider a mobile agent that performs a (trivial) management function, that is, a management station S dispatches an MA that visits all the devices in a managed domain. From each device r_i, where $1 \leq i \leq k$, it retrieves a set of managed objects. After visiting the final managed device r_k the MA returns to S. The notation below describes the travelling behaviour of the mobile agent. O_i is the set managed objects on each managed device that need to be retrieved from r_i :

$$O_i = \{o_{i1}, o_{i2}, ..., o_{iv}\} \tag{1}$$

where v is the number of managed objects to be retrieved from the device. The network transaction$\{r_{i-1}, r_i : O_i\}$ denotes the retrieval of the set of objects O_i from r_i and represents the migration of the MA from managed device r_{i-1} to r_i. The MA itinerary for the retrieval of managed objects from the domain of managed devices is expressed as:

$$\begin{aligned}\overrightarrow{O}_{MA} = \{\{S, r_1 : O_1\}, \{r_1, r_2 : O_2\}, ...,\\ \{r_{k-1}, r_k : O_k\}, \{r_k, S : O_S\}\},\end{aligned} \tag{2}$$

where $\{S, r_1 : O_1\}$ represents transmission of the initial mobile agent from the management station to the first managed object and $\{r_k, S : O_S\}$ represents the return of the MA (with all $k \times v$ lots of managed objects) from the last managed device r_i to the management station S.

The events denoted by the terms in $\overrightarrow{O}_{MA}$ occur sequentially, that is the event $\{r_i, r_{i+1} : O_{i+1}\}$ does not start until the event $\{r_{i-1}, r_i : O_i\}$ has completed.

The function $d_{MA}(\phi, s, i)$ is the network delay incurred by an MA travelling from the *ith* device to the $i + 1^{th}$

device (in order to retrieve the data from v managed objects stored on device i):

$$d_{MA}(\phi, s, i) = \phi + \frac{ma_{INIT} + ma_{\Delta}(i)}{s} \tag{3}$$

The term ma_{INIT} is the initial size of the MA dispatched from the management station. In this particular scenario the MA consists of code only and (as yet) no data. We choose a value $ma_{INIT} = 5$ Kbytes, which is based on values used previous MA performance analysis research carried out in [23]. The function $ma_{\Delta}(i)$ returns the amount of data the MA has collected up to and including the *ith* device and is given by the expression:

$$ma_{\Delta}(i) = 4vi \tag{4}$$

The function $ma_{\Delta}(i)$ is implemented specifically to our application in that v managed objects are retrieved by the MA. The number of managed objects is arbitrarily set to $v = 10$ and each object is assumed to be a 4 byte integer (thus the size of the MA increases by 40 bytes each time it visits a managed device). The parameters ϕ and s are the network propagation delay and network speed respectively. The expression is implemented this way so that it can be used generically to model both local area (LAN) and wide area (WAN) network environments. For example if we say that $\phi = \phi_{LAN}$ and $s = s_{LAN}$, where ϕ_{LAN} and s_{LAN} are the LAN propagation delay and the LAN speed respectively, then $d_{MA}(\phi_{LAN}, s_{LAN}, i)$ yields the network delay for an MA being transmitted from device i to device $i + 1$ over a local area network. Similarly $d_{MA}(\phi_{WAN}, s_{WAN}, i)$ represents the WAN case. The total time incurred from when the management station dispatches the initial mobile agent to reception of the MA from the last device in the domain k is given by the function:

$$D_{MA}(\phi, s, k) = \sum_{i=0}^{i=k} d_{MA}(\phi, s, i) \tag{5}$$

We set the parameter $k = K$, where K is the number of managed devices in the domain.

In our delay model we assume TCP (over IP) is the transport protocol that MA uses. However, we do not explicitly consider protocol overhead, such as packetisation, connection setup and termination or slow-start.

In this scenario, the management station is attached to a *local* LAN, while the managed devices are connected to a remote LAN. Both LANs are connected via a WAN using two (IP) routers (as shown in Fig. 4).

When the management station dispatches the initial MA, it is transmitted across the local LAN (to which the management station is attached) to the first managed device on the remote LAN.

After visiting the first managed device, the MA visits the remaining devices in the management domain. All transactions *between* managed devices are entirely contained within the remote LAN environment to which they are connected. Upon visiting the last device, the MA returns to the management station S when it has completed its retrieval of managed objects O_k on device r_k.

When an MA is transmitted between the management station and a managed device, it incurs two LAN propagation delays (one for the local and one for the remote) and one WAN propagation delay, So the total end-to-end propagation delay is given by:

$$\phi_{end} = 2\phi_{LAN} + \phi_{WAN} \tag{6}$$

Communication between management station on the local LAN and managed devices (first and last) on the remote LAN is rate limited by the speed of the WAN s_{WAN}, so the delay incurred sending the initial MA is given by $d_{MA}(\phi_{\xi}, s_{WAN}, 0)$. Similarly, the time taken to send the last MA is $d_{MA}(\phi_{\xi}, s_{WAN}, k)$.

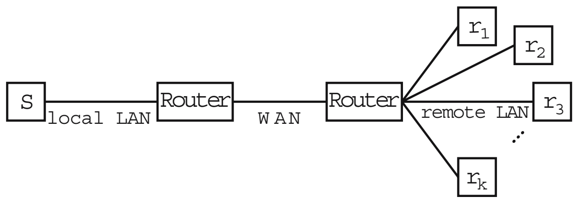


Fig. 4 Wide area network

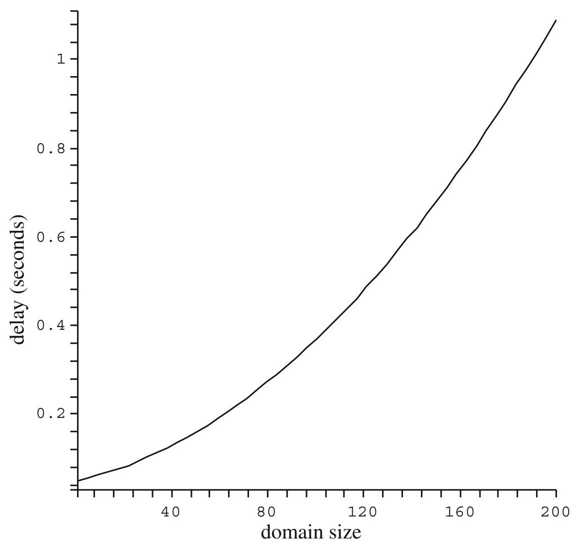


Fig. 5 MA delay for domain size $1 \leq K \leq 200$

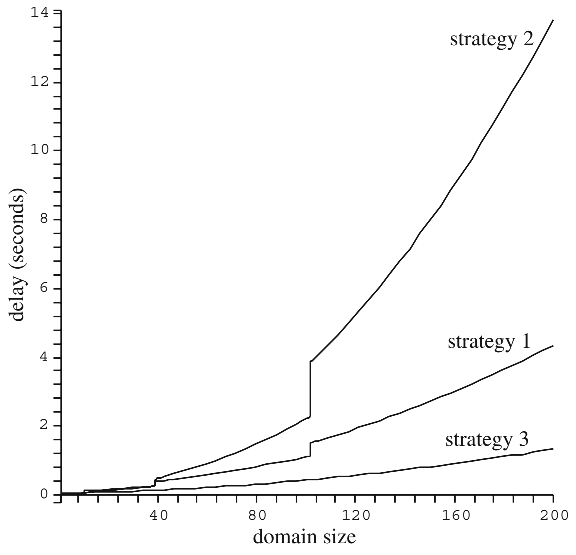


Fig. 6 A delay for strategies 1, 2 and 3, where $1 \leq K \leq 200$ for $y = 0.3$

Assuming router delay is negligible, the total MA delay is:

$$\begin{aligned}\xi_{MA}(k) = d_{MA}(\phi_{end}, s_{WAN}, 0)+ \\ d_{MA}(\phi_{end}, s_{WAN}, k)+ \\ D_{MA}(\phi_{LAN}, s_{LAN}, 1, k-1)\end{aligned} \tag{7}$$

In order to carry out an analysis of MA performance, we need to set some network parameters. A WAN propagation delay of $\phi_{WAN} = 20ms$ was chosen. The LAN propagation delay is $\phi_{LAN} = 17\mu s$ based on [24]. The WAN speed is $s_{WAN} = 200$ Kbytes/s (approximate speed in bytes/s of a 2.048 Mb/s E1 link) and the LAN speed is $s_{LAN} = 1$ Mbyte/s (approximate speed in bytes/s of a 10 Mb/s Ethernet). The graph in Fig. 5 shows the end-to-end delay $\xi_{MA}(K)$ for an MA to visit a domain for managed devices $1 \leq K \leq 200$.

In this analysis it was assumed that managed devices have of uniform dependence, thus an MA will visit each managed device the equal frequency.

We continue this analysis based upon a domain of managed devices having non-uniform dependencies. It is assumed that the dependencies are distributed according to a Zipf distribution. Zipf distributions were used to characterise the use of words in natural language [25]. That is the popularity of words plotted against their frequency of use was found to be distributed as a power law. The presence of this phenomenon has been discovered in many areas of research. One area in particular, that has been widely reported, is the relationship between popularity of Web sites and their access frequencies [26]. While we can present no evidence that such a relationship exists between the ranking of a managed device and its dependency metric for a particular manufacturing process, we argue it is a reasonable assumption to make. Though we accept that dependencies associated with managed devices may be arbitrary and not actually follow a Zipf distribution, we simply make the assumption in order to generate relative frequency figures for the managed devices within a cluster in order to apply our analysis.

For a domain of managed devices, if we place the managed devices in rank order from highest to lowest dependency, then the distribution follows the power law j^{-y} where and y is constants. We set and define the function which gives the dependency *value* of the *jth* most dependent managed device in domain:

$$\rho(y, j) = j^{-y} \tag{8}$$

Given that the managed device with the lowest dependency (k^{th}) needs to be visited at least once by an MA in any given polling period, then the visit frequency of any managed device j can be calculated:

$$f(y, j, k) = \left\lfloor \frac{\rho(y, j)}{\rho(y, k)} + 0.5 \right\rfloor \tag{9}$$

where the floor function x, yields the largest integer not greater than x (we add 0.5 for rounding). Clearly, if the MA visits every managed device in the domain at the frequency of the managed device with the lowest dependency, then the data collected will not reflect the state of higher ranked managed devices. We call this strategy 0, which despite keeping MA network traffic (and delays) to a minimum, is an inadequate method for collecting managed objects from higher dependency managed devices at the appropriate frequency. One solution to this would be to visit all k devices at the frequency of the highest ranked dependent device (rather than the lowest). We refer to this sampling method as strategy 1. The effect of deploying strategy 1 is that the management station sends out $f(y, 1, k)$ MAs,

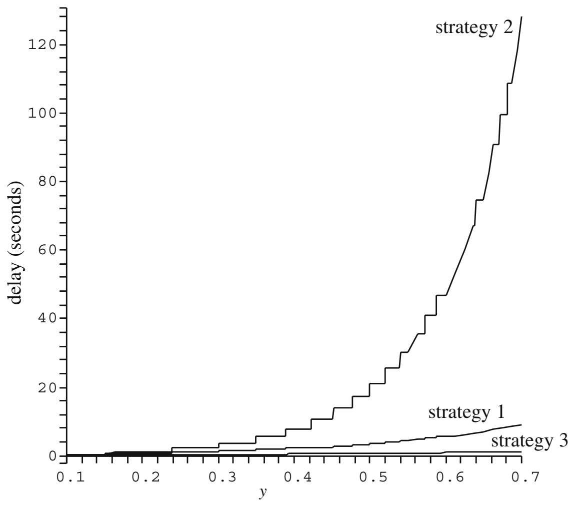


Fig. 7 MA delays for strategies 1, 2 and 3 for a domain size of $K = 100$ and $0.1 \leq y \leq 0.7$

rather than $f(y,k,k)$ MAs (in strategy 0), yielding an end-to-end delay $f(y,1,k)\times\xi_{MA}(k)$.

A second strategy (which we call strategy 2) would be for the management station to send out one MA *programmed* to visit every managed device in the domain before returning to the management station. With strategy 2, the MA delay is given by $\xi_{MA}(k\times f(y,1,k))$.

A possibly more efficient strategy, however, would be to visit managed devices at a frequency relative to its dependency. We call this strategy 3. With this strategy, the MA has to make a total of $F(y,k)$ visits to managed devices, where:

$$F(y,k)=\sum_{j=1}^{j=k} f(y,j,k) \tag{10}$$

For this strategy, the end-to-end delay is $\xi_{MA}(F(y,k))$. The graph in Fig. 6 shows the MA delays for all three strategies with $k=K$ (where $1\le K\le 200$) and $y=0.3$. Strategy 1 outperforms strategy 2, primarily due to the growth of the (single) MA making a high number of (repeated) visits to managed devices. Strategy 1 caps MA sizes by sending multiple MAs that make a lower number of visits. Even though multiple MAs that have to be dispatched by the management station, delays are significantly lower. This is related to the results in [1] where it was shown that methods of *clustering* could bring about performance gains in MA performance. However, strategy 3 yields the lowest performance because the frequency of visits to managed devices is proportional to their dependency.

Figure 7 shows the MA delays for: $f(y,1)\times\xi_{MA}(K)$, $\xi_{MA}(K\times f(y,1,K))$ and $\xi_{MA}(F(y,K))$, where $K=100$ and $0.1\le y\le 0.7$. It can be seen that, particularly for high values of y it is imperative to use strategy 1, or strategy 3. While strategy 3 yields the lowest delay, it would require MAs with a greater degree of sophistication. This would mean the MAs implementing strategy 3 would need more code and would therefore be larger in size. It would also mean more overhead with regard to processing, a factor we do not take into account in this analysis, but one that is likely to have a significant effect on delay.

6 Conclusions and future work

Contemporary manufacturing enterprises are facing ever increasing challenges. These challenges have led to a growing expectation that modern manufacturing enterprises should collaborate increasingly with their customers, suppliers and competitors in order to produce a supply chain. Organisations need to deal with a range of processes and activities, including the management of production scheduling, order processing, new product design, inventory management, warehousing, customer services and total quality control. In order to integrate these processes and activities, the manufacturing enterprise needs to rely on intensive information technology (IT) and networking techniques. An integrated manufacturing enterprise framework based on mobile agent technology brings flexibility and scalability to manage such a complex manufacturing networking environment. However, performance improvements can be achieved if mobile agent methods are facilitated strategically.

A process-driven perspective has been adapted to analyse a manufacturing environment because a manufacturing system consists of highly interrelated elements or processes. The management system needs to integrate the ever increasing number of applications and services with the corporate manufacturing enterprise's processes in order to yield maximum benefits. Therefore, dependency management was proposed in the field to capture and analyse work dependencies between processes at the engineering level and resources on the system level. An integrated framework connecting dependency applications is demonstrated in this paper.

We carried out an analysis of certain management processes which were dependent upon the managed devices within a domain with varying degrees of dependency. The implication of dependency ranking is that devices of a higher rank need to be monitored more frequently than lower ranked devices. If all devices in the domain are monitored at a frequency according to the lowest ranked device, then the gathered data will not accurately reflect the state of the system. On the other hand, if all of the devices are monitored at a frequency according to the highest ranked device, then undue load will be placed on the network. Furthermore, the delays introduced by unnecessary visits to lower ranked devices may be too great to achieve the necessary monitoring frequency. Our analysis shows that if each device is monitored at a frequency relative to its rank, then significant performance improvements can be achieved over monitoring devices uniformly at the frequency of the highest ranked device.

However, we would have to take into account the affects of the required increases in MA complexity in order to carry out more sophisticated *travel itineraries*. Increases in MA size and CPU consumption could negate performance gains brought about by reducing frequent visits to low dependency devices. We argue that such policy decisions will be based upon the nature of dependency *distribution* of devices. That is, for low values of y (assuming a Zipf distribution) we could consider dependencies to be distributed uniformly and be serviced by a simple travel itinerary implemented by strategy 1. However, the assumption of (approximate) uniformity does not hold as y increases thus we would consider more sophisticated travel itineraries.

Implementing a project requires a large set of processes to work collaboratively. Thus, it becomes extremely difficult to manage the dependencies between relevant elements and its supported elements. Thus, a well-designed decomposition approach for the analysis of manufacturing processes will bring high accuracy and efficiency of such a framework.

References

1. Huang C (2003) A strategic management framework for improving performance in a distributed manufacturing enterprise environment, Dissertation, Leeds Metropolitan University, UK
2. Chen W-SE, Leng C-WR (1997) A novel mobile agent search algorithm. Proc First International Workshop, MA'97, Berlin, Germany, pp 162–173
3. Brewington B, Gray R, Moisumi K (1999) In: Klusch M (ed) Mobile agents in distributed information retrieval, in intelligent information agents. Springer, Berlin Heidelberg New York, pp 335–395
4. Moisumi K, Cybenko G (1998) The travelling agent problem. Math Control Signals Syst 14(3):213-232
5. Dakin R (1997) The travelling salesman problem. http://www.pcug.org.au/dakin/tsp.htm, Accessed date: 31 December 2001
6. Jin Y, Zhao L, Raghunath A (1999) Activeprocess: a process-driven and agent-based approach to supporting collaborative engineering. Proc DETC'99-ASME Design Engineering Technical Conferences, Las Vegas, NA
7. Kar G, Keller A (2001) An architecture for managing application services over global networks. Proc Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2001), Anchorage, AK, USA
8. Keller A, Kar G (2000) Dynamic dependencies in application service management. Proc 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, NA, USA
9. Bagchi S, Kar G, Hellerstein J (2001) Dependency analysis in distributed systems using fault injection: application to problem determination in an e-commerce environment. Proc 12th International Workshop on Distributed Systems: Operations & Management (DSOM2001), pp 151–163
10. Brown A, Kar G, Keller A (2001), An active approach to characterising dynamic dependencies for problem determination in a distributed environment. Proc 7th International IEEE/IFIP Symposium on Integrated Management (IM2001), Seattle, WA, USA, IEEE Press
11. Hasselmeyer P (2001) Managing dynamic service dependencies. Proc 12th International Workshop Distribution, Systems: Operations and Management, Nancy France, Oct 2001, pp 141–150
12. Kim H-W (2000) Business process versus coordination process in organizational change. Int J of Flex Manuf Syst 12:275–290
13. Jain KA, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
14. Wu J, Cobzaru M, Ulieru M, Norrie D (2000) SC-web-CS: supply chain web-centric systems. Proc IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2000), Banff, pp 501–507
15. Xu Y, Brennan RW, Zhang X, Norrie DH (2000) A genetic algorithm-based approach to holon virtual clustering. Proc World Multi-Conference on Systematics, Cybernetics and Informatics (SCI, 2000), Orland, Florida, USA, pp 380–385
16. Kar G, Keller A, Calo S (2000) Managing application services over service provider networks: architecture and dependency analysis. Proc NOMS
17. Sabbaghian N, Eppinger SD, Murman E (1998) Product development process capture and display using web-based technologies. Proc IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, USA, 11-14 Oct. 1998, pp 2664–2669
18. Sullivan KJ, Griswold WG, Hallen B, Yuanfang Cai (2001) The structure and value of modularity in software design. Proc Joint International Conference on Software Engineering and ACM SIGSOFT Symposium on the Foundations of Software Engineering, September, Vienna
19. Svensson D, Malmstrom J, Pikosz P, Malmqvist J (1999) A framework for modelling and analysis of engineering information management systems. Proc 1999 ASME Design Engineering Technical Conferences, Las Vegas, NA, USA
20. Danilovic M, Borjesson H (2001) Participatory dependence structure matrix approach. Proc Third Dependence Structure Matrix (DSM) International Workshop, Massachusetts Institute of Technology (MIT), Boston, Cambridge, USA
21. Stamatelopoulos F, Chiotis T, Maglaris B (1995) A scalable, platform-based architecture for multiple domain network management. Proc IEEE International Conference on Communications (ICC'95)
22. Chase RB, Aquilano NJ, Jacobs FR (2001) Operations management for competitive advantage. McGraw-Hill, New York, USA
23. Adhicandra I, Pattinson C, Shaghouei EM (2003) Using mobile agents to improve performance of network management operations. Proc 4th Annual Symposium of Postgraduate Networking Conference (PGNET'03), EPSRC, Liverpool John Moores University, Liverpool, UK, 16-17 June 2003, ISBN: 1-9025-6009-4
24. Partridge C (1994) Gigabit Networking. Addison-Wesley, Reading, MA, USA
25. Nielsen J (2003) Zipf curves and website popularity. http://www.useit.com/alertbox/zipf.html, Accessed date: 20th August, 2003
26. Baldi P, Frasconi P, Smyth P (2003) Modeling the internet and the web-probabilistic methods and algorithms. Wiley, West Sussex, England