Coordination Approaches and Systems – Part II: An Operational Perspective

Graham Coates[1], Robert Ian Whitfield[1], Alex H. B. Duffy[2] and Bill Hills[1]

[1]Engineering Design Centre, University of Newcastle upon Tyne, Tyne and Wear, UK; [2]CAD Centre, University of Strathclyde, Strathclyde, UK

Abstract. *This is the second of two papers surveying research in coordination approaches and systems. This paper is concerned with operational coordination, which is aimed at coordinating activities such that the design process can be performed in a near optimal manner with respect to time, and the allocation and utilisation of resources. Aspects of coordination categorised as operational include resource management, scheduling and planning. The first of these two papers presents a review of coordination from a strategic perspective, which is concerned with the decision management aspects of coordination. Greater emphasis is now being placed on the significance of organising the design process as this affects time to market, product quality, cost, and consequently product success. The aim of this paper is to present a fundamental review of operational coordination approaches and systems. The 1990s has seen much progress being made towards a greater understanding and appreciation of coordination in various disciplines through the development of a wide range of approaches and systems. However, there remains a requirement to formally identify the key issues involved in coordination such that a widely accepted representation can be agreed upon. Consequently, research should continue to be supported in the exploration for a unified approach to coordination which will permit a broader and greater understanding of those aspects involved.*

Keywords: Coordination; Monitoring; Planning; Resource management; Scheduling

1. Introduction

Coordination has generated great interest and been an area of much research over recent years within a variety of disciplines. Indeed, coordination continues to gain prominence and be a focus of considerable attention. Engineering design, computer science and organisation theory are some of the disciplines that have attached significant importance, and consequently performed substantial research, in this field. The application of certain aspects of the various approaches to coordination surveyed in this paper could be applied equally as well to any discipline.

Presently, coordination remains ill-defined despite the significant amount of work carried out to date resulting in the proposal and development of a wide range of approaches and computer-aided systems. Each of these are predominantly aimed toward the implementation of the respective author's interpretation of what constitutes coordination and/or a coordination mechanism. In reality, a plethora of approaches and computer-aided systems now exist which generally only address individual aspects of coordination in isolation. Few of these endeavour to deal with more than one aspect of coordination simultaneously.

The primary objective of this paper is to survey existing approaches and systems related to the field of operational coordination and to identify the key issues addressed. In addition, common areas of research will be highlighted. Finally, a number of conclusions will be drawn from the survey.

The survey presented in this paper is intended to give an indication of the areas of operational coordination that have been, and currently are, being researched. It should also be noted that the survey is in no way exhaustive but is intended to be representative of the field of operational coordination. Whitfield et al. (1999) provide a parallel survey of approaches and systems from the perspective of strategic coordination in the first of these two papers.

The paper is arranged into three main sections. First, general approaches to coordination are discussed, which includes views and frameworks, etc. Secondly, resource management is surveyed including resource monitoring. Finally, scheduling and planning are discussed. While it is recognised that the approaches and systems surveyed could be

Correspondence and offprint requests to: G. Coates, Enginering Design Centre, Armstrong Building, University of Newcastle-upon-Tyne, Newcastle-upon-Tyne NE1 7RU, UK. Email: graham.coates@ncl.ac.uk

applied to a number of disciplines, the majority covered in this paper are concerned with computer-based systems. A summary of features of those coordination systems surveyed in the paper is presented in Table 1 in Appendix A.

2. Operational Coordination

Coordination, in an operational sense, with respect to completing tasks can be viewed as comprising of five fundamental components: activity, agent, order, location, and time. First, within any environment, in order to satisfy a particular requirement an activity needs to be performed so that the appropriate task can be completed. The activity needs to be specified such that when it is performed it will have the desired effect and complete the task. Careful consideration needs to be given to determine which activity is most appropriate to carry out in order to fulfil the task. Secondly, to perform an activity an agent, or agents, must carry out the required actions in order to complete a particular task. An agent can be considered as a resource and may be human, software or hardware. In simple terms, an agent is an entity capable of performing some activity to complete a given task. The correct choice of agent, or agents, will ensure that the activity is performed in the most suitable fashion and the task is completed satisfactorily. Thirdly, since relationships may exist between tasks, there may be an optimal order in which activities should be performed to complete the tasks. Consideration to task dependencies will enable the identification of those activities that can be performed concurrently and those that must be carried out sequentially. Fourthly, when an agent is performing an activity it may be appropriate to do so in a certain location. This consideration may be of particular importance and relevance when agents are working in the same team, or related teams, to complete the same task, or related tasks. Finally, for any activity, timeliness is usually of paramount importance. The time at which an activity is performed directly affects the completion of a task, and in addition may influence that of others. Hence, timeliness is a significant factor when considering coordination.

In summary, operational coordination is viewed as the concept of the appropriate activities being performed, in a certain order, by a set of capable agents, in a fitting location, at a suitable time, in order to complete a set of defined tasks.

2.1. General Approaches

A number of the papers surveyed discuss coordination as a single concept rather than dealing with individual issues. This section discusses those papers and categorises them into views and frameworks.

2.1.1. Views

Research into coordination is said to have attracted attention from a variety of disciplines during the 1990s (Malone and Crowston 1994). Thus, Malone and Crowston present an interdisciplinary study of coordination. The focus of much work is reported as being directed at coordination in parallel and distributed computer systems, human systems, and a combination of them both. Coordination is defined as the process of managing dependencies between activities. It is noted that this definition is influenced by a number of papers produced in the late 1980s. With respect to the definition, the requirement for identifying types of dependencies and coordination processes is recognised. The focus of coordination is placed on process management since a number of basic coordination processes are defined as managing: (a) shared resources, (b) producer/consumer relationships, (c) simultaneity constraints, and (d) task/subtask dependencies. Coordination mechanisms to manage these dependencies include goal decomposition, resource allocation and synchronisation.

Coordination and dependencies are important issues within organisational studies (Crowston 1996). The definition of coordination offered by Malone and Crowston, which states that it is the process of managing dependencies between activities, is adhered to and a taxonomy of dependency types and associated mechanisms is presented. Dependency types are stated as (a) dependencies between a task and a resource, (b) dependencies among multiple tasks and resources, and (c) dependencies between tasks or between resources. Coordination mechanisms include resource allocation.

Greenwood refers to Malone and Crowston's statement that coordination theory is the interdisciplinary study of coordination (Greenwood 1995). Providing a set of coordination abstractions, based on various disciplines, is seen as an aim of effective coordination theory.

A requirement for coordination exists within contemporary organisations (Greenwood et al. 1997). This need for coordination is caused by the increased complexity of many people and software tools working across various disciplines. Consequently, the coordination layer is presented as a device to manage dependencies between tools, people

and tools, and people. Tools are grouped in a common framework which is said to manage dependencies between them.

Coordination is proposed as a distributed search problem, where the search space provides a common representation for organisations, plans, and scheduling (Durfee 1993). This approach is stated as being the basis for an interdisciplinary study of coordination from computational and social perspectives. Durfee acknowledges that coordination has been studied from a number of discipline viewpoints. While alternative approaches have been proposed within the various disciplines, it is observed that the final objective is the same in each case.

The effects of runtime coordination strategies are considered within static organisations (Durfee and So 1997). These strategies are viewed as re-structuring the configuration of an organisation, thus affecting the performance of the agent system. Agents are described as human or computational. It is recognised that incomplete tasks can result in an uneven workload in an organisation and thus lead to reduced performance. A runtime coordination strategy is described as ensuring that, under normal conditions, agents work on complimentary tasks, however, in the event of agent failure, the surviving agents are applied most effectively. Role reallocation and local task reordering are named as two main coordination strategies. The response time and reliability of an organisation, and a combination of the two, are identified as the performance metrics of both these strategies. It is accepted that runtime coordination mechanisms have an associated cost which may outweigh its benefits.

Design coordination is defined as a high-level concept of the planning, scheduling, representation, decision making and control of product development with respect to time, tasks, resources and design aspects (Duffy et al. 1993). The relationship between design coordination and concurrent engineering is discussed. As a result, concurrent engineering is viewed as one way in which design coordination can achieve improved product quality, reduced cycle times and low unit costs. Consequently, design coordination is viewed as the vehicle for the realisation of concurrent engineering.

In situations where communication is either impossible or unreliable, the coordination of teams of agents can be problematic (Huber and Durfee 1995). In particular, the problem considered is that of plan recognition, which involves the idea of agents representing their commitment toward a joint mission. Agents must monitor others and assess whether or not the joint goal is still being pursued. Monitoring is said to enable agents to remain aware of collective commitment when communication is rarely possible.

Coordination and cooperation are two major concerns in systems involving distributed computers, termed intelligent decision making agents, sharing information and resources in order to solve a common set of tasks (Findler and Endler 1995). A distributed approach is said to offer a number of advantages for problem solving such as faster response and resource sharing. The problem considered is that given a set of tasks and a group of geographically distributed agents with limited resources available, how is task assignment handled and how are resource conflicts resolved. A negotiation technique and hierarchical iterative conflict resolution have been developed to solve the task assignment problem and the resource conflict problem respectively. Fundamental methods of task assignment involve each agent's evaluation of their ability and availability to accomplish tasks. It is said that agents from many domains are concerned with solving tasks that are geographically closest to them. Task evaluation is a quality measure, which is represented as a function of timeliness, primary resource cost factor, support resources factor, and workload factor. The negotiation strategy used to determine task assignment involves agents assessing their ability to complete tasks using the task evaluation quality measure and then exchanging these with one another. Agents with the smallest quality measure are said to become self-appointed Commanders for those respective tasks. Further criteria are used to decide this issue if agents have equivalent quality measures. Once all tasks are assigned Commanders, every agent commences solving their particular tasks. Hierarchical iterative conflict resolution is used for resolving conflicts regarding resources. Conflicts are solved based on task priority, that is agents with higher task priorities borrow resources from other agents. It is noted that while agents resolve conflicts, other agents must not be idle as this would diminish the benefit of distributed agents.

2.1.2. Frameworks

The Design Coordination Framework is proposed as a means of supporting the coordination of product development (Andreasen et al. 1996). Eleven frames are presented and are said to be important contributions for monitoring the design activity. The shift from traditional design models to concurrent engineering is recognised. It is highlighted that the concurrent engineering perspective does not recognise that the key to achieving optimal design performance is the effective coordination of the design process.

That is, activities should not necessarily be performed in parallel but rather organised to achieve optimum performance.

Within the Concurrent Simultaneous Engineering Systems (CONSENS) project, techniques related to coordination such as release control and change management are discussed (Bullinger and Warschat 1996). Release control is described as involving the management of information that is susceptible to continuous alterations as the product evolves. Change management is said to manage the product change process by keeping track of the changes that need to be made, including why and when they are to occur.

Intelligent agents need to interact with each other to cooperatively achieve their objectives (Durfee 1993). This contrasts with the popular concept in Artificial Intelligence that intelligent agents should act by themselves. Distributed Artificial Intelligence approaches to coordination are said to be based on three social metaphors, namely organisational theory, planning and scheduling. A single framework, called partial global planning, is cited as a framework that unites these approaches and has been designed to coordinate agents that are cooperatively solving problems. This is said to involve coordination of both sharing tasks and results, adhering to long-term organisational roles and reactively planning to achieve short-term objectives. Organisational structure is recognised as having an affect on coordination quality and overhead. As the number of agents increases, the choice of organisational structure can impact the complexity of coordination. Experiments involving partial global planning have been shown to support this fact.

Tan et al. (1996) propose a multi-agent framework to develop design and planning using the concurrent engineering approach. The objective of the framework is declared as improving initial product design in terms of feasibility and manufacturability with respect to time and plan constraints. A blackboard architecture and an intelligent agent network are included in the framework, which is aimed at representing a design team by modelling a team member's viewpoint as a segment of continuum knowledge. The electronic blackboard handles the control and coordination of the system, in addition to holding a list of outstanding tasks and scheduling which task is to be performed next on a priority basis. Intelligent agents are software programs, which perform functions such as conflict detection, simulation, and cooperate with humans to solve problems. Team members, which are described as flanking the network of intelligent agents, are individuals in the problem-solving group. Each team member is able to access a central archive, which contains all knowledge bases of the various domains. The information system technique is described as an effective means to shorten the product development cycle.

The Coordination Signal Framework involves domain specific agents orchestrating domain independent agents (de Jong 1997). A domain independent agent is said to learn to coordinate its actions by relating the coordination evaluation signals it receives from the domain specific agents to the environment. Thus, the domain independent agents are said to behave usefully in an unknown environment. This feature of a particular type of agent only being able to receive signals is different from the more common case of structured information, in the form of messages for example, being exchanged between agents. An application of the Coordination Signal Framework is reported as illustrating that an agent that can only listen for signals can coordinate its actions in relation to agents that are able to send and receive signals.

Design coordination is viewed as involving organisation and control, integration, and modelling (Carter and MacCallum 1991). Furthermore, it has been recognised that there is a need for computer support for design coordination. The Hierarchical Object Oriented Blackboard System (HOBS) comprising of a number of components is presented as such a computer based support system. The Executive is the control mechanism, which elects the next action or operation to be carried out. Knowledge sources are arranged in a hierarchy and are said to be segments of design expertise or resources. Communication between knowledge sources about the product occurs through a Workspace. The Workspace is described as a general resource area for the system, which handles data, tools and messages. For each resource type, the Workspace comprises of the Database, the Toolbase and the Message Board. Communication links only exist between certain components. To permit those components to interact with one another, message passing is used. An application of HOBS is stated as showing considerable benefits over current practice.

Support tools can be used to aid distributed humans and computational agents in order to coordinate their activities by assisting the management of their agenda (Decker and Lesser 1995). A User Coordination Assistant Agent (UCAA) and an Agent Coordination Module (ACM) have been developed as components of such a support tool. Both the UCAA and ACM consists of a number of modules. The UCAA is situated at a person's workstation keeping track of current tasks and offering task schedules according to supplied preferences. Each person has a UCAA which

is linked to all other UCAAs such that a distributed coordination process occurs and agendas are produced in a collaborative manner. An ACM can provide agenda management to coordinate computational agents according to a human's task-order preference. Problems are said to be solved in a timely manner and an efficient way that dynamically adapts to the current situation.

The Contract Net Protocol involves the communication among nodes in a distributed problem solver, and is described as a high level protocol facilitating the control of cooperative task execution (Smith 1980). Speed, bottleneck avoidance and reliability are identified as motives for distributed problem solving. The specific problems addressed by the Contract Net Protocol is that of task selection and allocation to nodes, knowledge source selection for task execution, and load balancing. Nodes are described as participating in contract negotiation in order to solve these problems. Negotiation is described as an interactive mechanism, which, amongst other things, achieves task distribution through distributed control, transfer of information, and resource allocation control. Nodes are stated as being dynamic throughout problem solving, and classed as either manager or contractor. Managers monitor task execution and process results whereas contractors execute tasks. Node communication is described in terms of six basic messages, which are said to capture the types of interaction that occur in a task sharing approach to distributed problem solving.

An agent-based system, named the Design Coordination System, is described in which software agents work in a coordinated and cooperative manner to enable the efficient completion of some computational analysis (Coates et al. 1999a, 1999b). Task and activity management, scheduling, and task execution are performed simultaneously while taking into account the variable nature of the computing environment. Coordination is said to involve five key issues with respect to completing tasks, namely activities, time, order, agent and location.

2.2. Resource Management

The use of many resources to facilitate the efficient performance of activities is an approach with obvious benefits. To maximise the utilisation of resources it is clear that some form of management, including monitoring, is required such that activities can be performed on the most suitable resources at an appropriate time. Resource management is an aspect of coordination that has received much attention during this past decade, whilst the concept of resource monitoring appears to have been less of a consideration. However, the late 1990s has seen more emphasis and recognition in the area of resource monitoring, particularly within parallel and distributed computing environments.

Resource management is considered in a shared memory multiprocessor where resources are most effectively utilised by using a single task queue for all processors (Yen and Bastini 1995). Idle processors are assigned tasks from the centralised task queue, so that system load is automatically balanced. The systems considered involve clients and servers, each of which runs on its own processor. Tasks are placed in the task queue by clients which are then removed by the servers and then executed. Two algorithms for decentralised resources management, which are described as robust and efficient, are presented. The fault-tolerant parallel task queue algorithm involves free processors coordinating their parallel access to the task queue. A tournament-ticketing algorithm has been developed to avoid task queue access contention, which involves a set of clients and a set of servers competing for tickets to gain access to the queue. System execution flow is managed using a locking mechanism to synchronise the client and server. The robust parallel access task web algorithm employs a web data structure to maintain the tasks in the system. Clients insert tasks into the web at the outer nodes and servers access the inner nodes of the web to locate tasks and execute them. This algorithm is considered to be more efficient than the previous since the tournament-ticketing algorithm is said to have a considerable time overhead when the number of processors is large. Both algorithms are said to achieve optimal reliability since the system continues computation providing at least one working processor exists.

A resource management repository coupled with appropriate support tools is said to have become a necessity in a distributed computing environment (Quinn 1994). In particular, integrating system measurements from heterogeneous platforms into a common resource information repository is required. Information held in such a repository could be used for planning future capacity requirements and assisting Information Systems Managers. A number of application areas are identified as having a need to integrate Unix systems into a resource management repository. Common requirements of these application areas are identified such as collection, organisation, storage, distribution, retrieval, and analysis facilities. The ability to measure Unix system resource utilisation and service at the global level,

application level, and process level are requirements of the resource management repository for Unix systems. At each of these levels, various measurements can be collected. In addition, measurements can be obtained on device activity, Local Area Network (LAN), and system configuration information. A structure for resource management repository from Unix systems is suggested, which incorporates those features mentioned.

The Distributed Resource Management System (DRMS) has been developed to support dynamic reconfiguration of data parallel applications in an environment of unpredictable and varying workload (Moreira and Naik 1997). The resource control and scheduling mechanism is said to coordinate the execution of re-configurable applications. A unique feature of the system is stated as closer coupling between application and resource coordination, and scheduling activities which are normally associated with the operating system. Re-configurable applications can be used since it is possible to dynamically change the number of concurrently executing tasks of an application and the resource allocation during the course of program execution. The reconfiguration mechanism of the DRMS is based on the Schedule and Observable Point (SOP) programming model in which parallel program execution consists of a sequence of stages. SOPs define stages, which consist of resource, data, control, and computation. This DRMS environment consists of a scheduling module that controls the processors of the parallel system. For each parallel job, the scheduling module creates a partition of processors that executes the job. This partition can be resized at the SOPs of that application, but is fixed for the duration of a particular stage. The scheduler also decides on the mapping of application tasks to processors in the partition. Only one task is mapped onto a processor in the current application. The DRMS has functional components with interactions among them. The DRMS compiler is said to translate programs with DRMS annotations, linking them with a Run-Time System (RTS) to create re-configurable application executables, which are the files that are actually run. The Resource Coordinator (RC) and the job scheduler and analyser (JSA) are responsible for resource coordination and task scheduling. Run-time management and coordination of user applications are achieved by the User Interface Coordinator (UIC), the RC, and the Task Coordinator (TC) and run-time monitor. The performance analysis component is handled by a run-time performance data gatherer, and associated tools and utilities. The performance gathering component is designed to assist users and system administrators, and provide feedback that the JSA can use to make more intelligent decisions. Once submitted using the UIC, applications are assigned a TC. TCs are described as consisting of a number of agents equivalent to the number of tasks within an application. Since the number of tasks may vary during application execution, then so does the number of TC agents. TCs communicate via sockets and are said to acquire/release processors from/to the RC and start/restart application task execution the allocated processors. Results indicate that reconfiguration of the various applications considered is effective under the DRMS, and can be beneficial in improving individual application response time and overall system utilisation.

Nguyen et al. (1996a) propose a technique to automatically regulate the number of processors used in the execution of an individual parallel program so as to maximise speed-up S. The speed-up S is defined as the time taken to solve a particular problem using a single processor T_1 divided by the time taken using a number n processorsT_n. That is, $S = T_1; / T_n$. It is recognised that the need for such a technique exists since speed-up does not increase monotonically with the number of processors for many parallel applications. The runtime system proposed, described as being able to dynamically adjust processor allocation, is reliant on appropriate hardware support to dynamically measure application efficiencies at different processor allocations. These measurements are used to calculate speed-up, and then an adapted method of golden sections optimisation technique is used to find the best allocation. A basic self-tuning algorithm is described, which uses initial measurements as being valid for the complete execution duration of an application. An obvious deficiency of such an algorithm is that measurements may in reality change during the course of the execution. Thus, extensions of the algorithm are also described. A change-driven self-tuning algorithm is said to continuously monitor job efficiency and repeat the search procedure whenever a significant change in efficiency is noticed. Since job efficiency could change during the self-tuning search but stabilise before the search completes, the algorithm can still settle on an incorrect allocation. Therefore, a further extended algorithm, termed time-driven self-tuning, includes change-driven self- tuning and also repeats the search procedure periodically, regardless of job efficiency. To assess the performance of self-tuning, a selection of representative parallel applications were run with no tuning, basic self tuning, change- driven self tuning, and time-drive self tuning. One main observations made was that performance can be

improved, however the cost of monitoring must not outweigh this benefit.

Distributed systems need resource management capabilities that can allocate resources to applications, monitor and control the use of resources, and reallocate resources due to anomalies (Davis and Sydir 1996). It is recognised that a resource management system should provide the suitable quality of service for each application and user according to their individual needs and the available resources, etc. Commercial products are said to be designed to manage networks rather than resources. A need is identified for research to develop new techniques that will manage system resources in a uniform and coordinated way within a dynamic environment while satisfying application quality of service requirements. Various aspects of quality of service are discussed with regard to resource management.

An agent is dedicated to the efficient employment of the available resources within a computing environment (Coates et al. 1999a, 1999b). The Resource Manager agent is described as being responsible for ensuring that the near optimal use is made of the available resources throughout the lifetime of the computational analysis. This involves maintaining and acting upon the information held within a resource model which changes with time due to the dynamic nature of the computing environment. If the resources are not being utilised to their potential the Resource Manager recognises this fact and instructs a Scheduling Agent to re-schedule using a multiple criteria genetic algorithm.

2.2.1. Resource Monitoring

In presenting a heuristic algorithm for design-to-real-time scheduling, the frequency of task execution monitoring is identified as an important criterion (Garvey and Lesser 1994). With regard to the problem under consideration, results indicate that monitoring almost always provides a reduction in missed deadlines. It is also suggested that monitoring may only have benefits at certain loads, and therefore monitoring rates should be reduced at other loads.

Nguyen et al. (1996b) consider the use of run-time measured workload characteristics in parallel processor scheduling. A main conclusion drawn from the work was that despite the inherent inaccuracies of runtime measurements, and the added overhead of more frequent reallocations, schedulers using them can significantly outperform those that do not.

Komodo, Network Status Predictor and *Network Weather Service* are three very recent resource monitoring systems of computer networks all of which use the technique of sample message sending between two nodes to determine the round trip time to establish network latency.

Resource monitoring is considered in the context of mobile programs and presented as being based on three hypotheses (Ranganathan et al. 1996). First, resource aware placement can vastly improve performance of constituents of a distributed application. Secondly, deciding which resource to run particular applications can be based on monitoring variations in network characteristics over the medium to long term, that is several minutes to a number of hours. Finally, a simple and inexpensive resource monitoring strategy can provide adequate information for effective mobility decisions. A distributed network latency monitor called Komodo has been implemented as a user-level daemon, which is a process that runs continuously in the background, on every host. Applications initiate monitoring requests to their local Komodo daemon, which consequently handle the request themselves or refer it to the relevant daemon. Program mobility decisions are based on the monitoring information obtained. An important issue regarding the design of a resource monitoring interface is identified as whether applications should poll or be notified asynchronously of resource changes. Some factors considered regarding polling or notification are the frequency at which applications need information and the associated cost. The concept of a threshold is discussed which involves only those resource changes being tracked that exceed the mentioned threshold. This approach is said to work well if resource level changes are stable whereas transient changes can cause spurious responses. The inclusion of a filter in the jitter-based scheme is proposed as a method of eliminating transients.

The Network Weather Service (NWS) is presented as a forecaster of resource performance in a meta-computing environment (Wolski 1997). Such a forecasting method is identified as satisfying the need to support parallel application scheduling. Resource contention is seen as a cause of load and availability variation. As a result, this is identified as contributing to the performance variation that resources can provide an application. The NWS is described as a distributed service that continually predicts network resource performance so that parallel application schedulers can adapt to load variations. Distributed sensors are described as periodically accumulating instantaneous conditions, which are used with numerical methods to project future conditions. Monitoring is described as non-intrusive since performance is not compromised by the service provided. The NWS forecasts network

performance, described as latency and bandwidth, as well as available Central Processing Unit (CPU) percentage for each machine being monitored. The NWS is said to sense resource performance, forecast future performance, and disseminate forecasts to the schedulers concerned. An NWS administrative client utility controls the system. NWS servers exist for each machine being monitored. Each server has a network performance sensor and a CPU availability sensor. The NWS utilises a number of forecasting methods once performance measurements from each resource are taken. Method selection is said to be dynamic since all are employed concurrently, and that with the lowest error measure is used to produce the forecast. It has been recognised that while resource scheduling in a distributed parallel computing environment has been the focus of much research over recent years, network monitoring has been largely neglected (Kim and Lilja 1998). It is understood that the dynamic varying nature of network load can substantially impact resource contention and performance. Consequently, the Network Status Predictor is presented and described as a distributed system that periodically detects and predicts network load. As emphasised by Wolski (1997), the NSP is proposed as a support to dynamic network resource scheduling. Network traffic is categorised into classes, and depending on the traffic class, one of a number of numerical models are used to predict future network load. As with Ranganathan's Komodo, a user-level daemon runs on each node attached to the network. Latency is periodically sampled by sending and receiving some fixed messages. A difference between NPS and Komodo is reported as that while both detect network latency, NPS also predicts future latency. In addition, while both NPS and NWS use prediction, NPS uses a prediction range of network latency whereas NWS is said to use a single number value.

Resource monitoring is identified as an integral requirement of coordination (Coates et al. 1999a, 1999b). The need to continuously assess the status of the available resources, and subsequently act on the information when necessary, is seen as imperative if the resources are to be used in the most effective manner. Agents termed Resource Monitors exist for each resource within a distributed computing environment, which continuously monitor, record and analyse the efficiency and status of their associated resource. If a deviation in a resource's efficiency is observed by a Resource Monitor then the Resource Manager is informed. This may result in the Resource Manager deciding to remove or add that particular resource from or to the design environment and request that a new schedule be calculated by the Scheduling Agent.

2.3. Scheduling and Planning

The papers surveyed in this section enforce the understanding that scheduling and planning is an important aspect of coordination. Many authors have identified and view scheduling as the basis for coordination. Indeed, in some cases scheduling is seen as coordination.

Design-to-time supports the use of all available time to produce the best possible solution, the success of which depends on the predictability of method duration and quality (Garvey et al. 1993). Thus, a design-to-time scheduler for complex task structures that include particular kinds of relationships between tasks is presented. Tasks are thought of as being inter-related, rather than the popular concept that they are either independent or have precedence constraints between them. Task inter-relations are defined as enable constraints, facilitate relationships, and hinder relationships. Design-to-time environments are described as those in which a number of methods are available to solve tasks. Methods make trade-offs in solution quality against execution time. The design-to-time scheduler employs pruning to improve efficiency, which is a technique to remove associated methods that do not generate greater or equal quality solutions in equal or less time for each task. From the number of schedules generated, that exhibiting the maximum quality possible without breaking any deadlines is maintained. An important attribute of the scheduler is that it can control its own performance by dynamically modifying the task structure being scheduled. However it is stated that the removal of methods may result in the best schedule not being found.

A heuristic algorithm for design-to-real-time scheduling is presented with the aim of maximising the overall solution quality of a given set of tasks in a stated amount of time, while missing as few deadlines as possible (Garvey and Lesser 1994). Tasks have a designated start time, deadline, estimated processing time, and set of methods with associated processing times and solution qualities. Methods consist of a set of subtasks, each of which can have a distinct earliest start time, and an estimated processing time. The architecture employed within the algorithm consists of a controller and execution subsystem. It is the execution subsystem that signals to the controller that an unpredicted event has occurred. The controller designs a high-level solution to a problem using all of

the available resources as efficiently as possible. A controller plans which tasks to perform, what solution method to use for each task, and when to work on each task, based on expected time to process each task, quality versus time trade-offs between alternative methods, and the distribution of deadlines. An execution subsystem then directs the actual execution of low-level problem-solving steps, potentially interleaving steps from several tasks. Feedback from the execution subsystem to the controller allows the rescheduling of tasks when necessary because of inaccurate predictions or unexpected events.

An approach to multi-agent coordination is presented consisting of a standard operating procedure and look-ahead coordination (Liu and Sycara 1996). The purpose of the standard operating procedure is to control task coupling and minimise communication. Look-ahead coordination is said to increase agent visibility and provide cues for decision adjustment. The environment considered involves agent's tasks being tightly coupled and requiring real-time scheduling and execution. Resources are assigned agents responsible for making decisions and monitoring resource usage. The standard operating procedure used is dispatch scheduling. Solution quality is said to suffer since agents make decisions based on local and current conditions. It is hypothesised that agent coordination, which extends agent visibility of problem solving conditions, can obtain higher quality solutions without compromising computational efficiency. The look-ahead coordination mechanism, called coordinated forecasts (COFCAST), operates on top of dispatch scheduling in order to improve performance and only adds a low overhead to the standard operating procedure. COFCAST increases agent visibility based on information regarding global and future conditions. Coordination is seen as the operational sequencing of agents so as to reduce the cost of the final schedule being late. To coordinate agent's forecasts, predicted ready time is assigned to each operation, which is dynamically adjusted during the scheduling process. The approach was tested in a real-time job shop schedule optimisation domain in order to assess agent forecast accuracy, agent interaction complexity, and availability of indicative information. The objective was to produce a schedule with minimised weighted tardiness. The coordination technique implemented was based on a blackboard model, where agents communicate information on a shared memory space. Experimental results show that the look ahead coordination significantly enhances the performance of the standard operating procedure in solution quality, and that the approach is capable of producing high quality solutions in a real-time environment.

A heuristic multi-agent planning framework is presented in which the overall goal of a group of agents is divided into subgoals from which subplans are derived (Ephrati and Rosenschein 1994). Agent's subplans are derived separately and in parallel, and are merged into a unified global plan. Agents communicate to construct this global plan. Since agent's subplans may conflict, an iterative search is employed to dynamically generate alternatives from which an optimum global plan can be identified. The approach claims to reduce overall planning time while deriving the optimal global plan that would have been derived, given those original subgoals. In multi- agent environments this approach is also said to remove the need for a central planner with global domain knowledge and of the agents involved. Two scenarios are considered in the paper, both of which are explained through examples. The first involves a group of agents with a static common goal. The second considers how agents can interleave planning and execution when planning towards a dynamic common goal. In both scenarios the approach is a cost driven merging process that results in a coherent global plan, given the sub-plans.

Altus et al. (1996) present a method for structuring tasks with optimal ordering and decomposition. The use of decomposition in current research is described as a way to simplify large complex problems. Computation time is said to be reduced since the scope for parallel computation is increased. It is identified that a decomposition tool must include a number of objective functions such as small groups, small feedback, and small information transfer. Genetic Algorithms (GA) are said to be a flexible method for optimal scheduling and decomposition, in comparison with the more traditional methods, since different objective functions can easily be compared. A GENetic algorithm for Decomposition and Analyses, AGENDA, is presented as a tool for enabling users to assign analyses of subproblems or order analyses subject to any number of criteria. It is indicated that while AGENDA was specifically aimed at planning decomposition on multidisciplinary optimisation problems, it is also useful in scheduling sequentially executed tasks to reduce feedback.

Jennings and Jackson describe the design and implementation of an agent-based distributed meeting scheduling system (Jennings and Jackson 1995). Intelligent agents, called Meeting Scheduling Agents (MSA), exist within the computer desktop of each user and have knowledge of their user's preferences and commitments. An algorithm is encoded in each

MSAs decision making module which is used to arrange meetings. The system is invoked when a user indicates to its MSA that it would like to schedule a meeting involving certain individuals at a given time. The user's MSA advertises the proposed meeting to the associated MSA of those invited users. When a proposed meeting conflicts with existing commitments of others, iterative negotiation is used to resolve the conflicts. Possible resolutions include cancelling or rescheduling the meeting, or rearranging existing meetings. This system is said to improve on its predecessors as it caters for postponement and rescheduling of meetings, and their effective scheduling in a dynamic environment. This improvement over current calendar management products is facilitated by agents, and not the humans, which manage and enact the negotiation process.

A software agent within a computer-aided coordination system is dedicated to scheduling tasks, or application executions, within a distributed computing environment (Coates et al. 1999a, 1999b). A multi-criteria genetic algorithm (Todd 1997) is employed by the Scheduling agent in order to produce a near optimal schedule of the outstanding tasks. The genetic algorithm can be used a number of times during some computational analysis depending on the variation of resource usage and efficiency. This approach of using a genetic algorithm in a dynamic fashion ensures that the objective of maximising the effective utilisation of the available resources is satisfied at all times.

2.3.1. PERT and CPM

Extensive literature exists on Program Evaluation Review Technique (PERT) and Critical Path Method (CPM). In the late 1950s, the US Navy first developed PERT as part of the Polaris Missile System Program. It was introduced for application in projects involving uncertainty. Three time estimations, for each project activity, were required to estimate the expected mean time, namely optimistic time, most likely time, and pessimistic time. CPM was developed in 1957 by DuPont, and is a deterministic approach where estimation of one time quantity is used for each activity and there is no statistical feature on uncertainty. CPM does include a mathematical procedure for estimating the trade-off between project duration and project cost. Briefly, PERT and CPM use network diagrams showing precedence relationships between a number of activities required to be performed to meet some defined objective. A procedure is then used to find the primary project parameters such as the critical path, the minimum project completion time, and the various leeway times.

Kamburowski (1985) presents an algorithm to obtain the upper and lower bounds of the expected project completion time. A recurrence method is used to determine the lower and upper bounds on the mean event occurrence times. It is recognised that the proposed method is simpler than other comparable methods and although the accuracy of the approximations are reduced, they are said to be acceptable.

The graphic tool X-PERT employs techniques for visualising the computer-aided development and analysis for PERT diagrams (Di Battista et al. 1989). X-PERT has an automatic layout capability which allows the schedule of large complicated projects to be viewed.

Ord (1991) presents a simple approach to PERT, which is proven to perform an acceptable degree of approximation. The method is said to provide an assessment of criticality for each activity in the network. A discrete approximation of activity time is explored rather than the classical technique of optimistic, likely and pessimistic activity time.

A simplified method to PERT/CPM is proposed that enables critical project parameters to be ascertained manually as opposed to using the standard computer algorithm (Zhu and Heady 1994). A five step procedure describing the simplified method of finding PERT/CPM critical paths, project durations, and slack times is presented. First, a table is set up to contain the time required to complete all activities before the next node. Secondly, the entries for the table are computed. Thirdly, the critical path is identified by backing up from the known ending point. Fourthly, free slack times are determined. The final step is finding the total slack times.

A shortcoming of the classical PERT approach and method is said to be that the path with the greatest expected duration is chosen as the critical path (Soroush 1994). Soroush challenges this definition by stating that the most critical path to be that in which the probability of completing its activities by a given time is less than that of every other path.

PERT is an established management tool, which can be used to identify relationships between various project milestones and determine the critical path of activities (Aikat 1996). Thus, PERT assists in identifying those areas of a project that require most resources. PERT is also described as a graphical representation of the project plan and schedule.

The importance of uncertainty can be used to identify activities that need more attention in reducing the magnitude of uncertainty of the project completion time (Cho and Yum 1997). Thus, a procedure for

evaluating the uncertainty importance measure of an activity, or a pair of activities, in a PERT network has been developed. PERT networks are classified as either having or not having a dominantly long path. The two-stage procedure based on the Taguchi tolerance design is aimed at PERT networks classified as not having dominantly long paths. The first stage involves screening out the activities with negligible main effects. The second stage estimates the main and/or interaction effects of activities, and their contribution ratios are used to estimate the desired uncertainty importance measure of activities.

Chan assesses the strength of simulation modelling in project management with limited resources (Chan 1997). It is stated that project scheduling decisions are generally subject to both precedence and resource constraints. PERT and CPM are cited as the most common network techniques for project scheduling. Chan explores the possibility of using simulation software, namely Simfactory 11.5. The simulation model allows an estimate to be made of the performance of an existing system under some projected set of operational conditions. Simulation results are claimed as not necessarily giving the optimum solution, but alternative system designs are proposed that can be compared to see which best meets specified requirements. Simfactory 11.5 is defined as a compliment to both PERT and CPM, and can work on deterministic and stochastic models. Two similar case studies are detailed which use Simfactory 11.5 to solve the project scheduling problem for both deterministic and stochastic scenarios. The deterministic scenario involves activities of constant time whereas the stochastic scenario uses a uniform distributed time, which can have a time variant added to every activity to account for uncertainty. The experiments are described as an alternative to PERT and CPM in dealing with any project management problems. Results indicate that Simfactory 11.5 is capable of managing projects with both precedences and resource constraints. The most important feature is stressed as that a resource constrained project, while taking into account the uncertainty and at the same time performing the time, cost analysis can be solved. In fact, it is stated that due to Simfactory 11.5 being able to compute the total project cost, and perform a sensitivity test, it is not impossible that simulation will replace PERT and CPM as a project management tool.

2.3.2. Project Management

Dellen and Maurer (1996) identify the main demands on the dynamic planning of development processes and present methods and techniques to meet them. Since development processes are characterised by high complexity and many people, projects have to be decomposed into interdependent tasks. Coordination of the activities and people is identified as requiring great effort. It is recognised that planning decisions may need to be modified since the design process can change. In addition, certain planning decisions can only be made as a result of the completion of particular stages of the design process. Based on these issues, Dellen and Maurer identify a number of demands on a project planning environment. Their work is described as developing methods and techniques for computer-based support of distributed planning and execution of design processes. The CoMo-Kit project is named as developing techniques, methods, and systems which support planning and execution of complex distributed cooperative development processes (Maurer 1996). A Modeler and Scheduler are provided for planning and execution software development processes. The Modeler describes project plans using tasks, methods, products, and resources. The Scheduler is said to interpret the models and manage the process information, necessary to give useful planning and execution support. Descriptions of what process information is managed and how it is managed by the Scheduler are given. REDUX is identified as the planning and design model for the implementation of the CoMo-Kit Scheduler. The Scheduler is described as extending REDUX, which supports task decomposition and dependency directed backtracking, by managing data flow dependencies. Different possibilities to refine, extend and adapt the process model during execution with CoMo-Kit are described. In particular, method selection, addition of new methods, model refinement, decision rejection, faulty model correction, and delegation retraction are reported.

Procura is a project management tool for planning and scheduling agent-based design projects (Goldman 1996). Dellen and Maurer's CoMo-Kit is cited as a workflow management tool that integrates planning and plan execution, but unlike Procura, it does not address the issue of scheduling. In Procura, planning, scheduling and plan execution are said to be interleaved. Planning is described as the division of tasks into subtasks. Scheduling is viewed as the assignment of resources and designation of task start and end times. Plan execution is stated as the assignment of values to design variables. The plan and schedule are incrementally revised during the design and progressively becomes more detailed. Procura is said to support the user with planning and re-planning, as well as scheduling and re-scheduling. Agents are humans or computer that use interfaces to

Procura. Workers are defined as humans with no link to Procura. Agents perform tasks using workers and equipment. It is the agents, workers and equipment that are scheduled to carry out the tasks of the plan.

Bendeck et al. (1998) consider the coordination of project planning and scheduling, and monitoring during execution. A system has been developed to address the issues of coordination and notification in the context of a distributed software development design process. A number of general and role-specific requirements that support the coordination of management activities in a software engineering environment are discussed. General needs include planning, scheduling, monitoring and notification. Role specific requirements include software process agents such as a project planner, measurement planner, project manager, quality assurer, and technical roles. A set of scenarios for distributed, cooperative software development are described from the perspective of each software process agents. A system architecture is described, which includes modelling and kernel components and an experience factory, along with the software process agents, which are able to handle the management activities during a software development project. The modelling component stores process, product, and quality models. The kernel component maintains the plan and schedule, and project's execution state. The experience factory stores general models and project data.

2.3.3. Dynamic Scheduling

Dynamic processor allocation policies are said to rely on determining job characteristics, such as job efficiency and speed-up, at runtime (Nguyen et al. 1996b). The value of runtime measurements is said to depend on issues such as measurement accuracy, parallel application stability, application performance estimation, and reallocation costs. Experiments are performed in both interactive and batch environments. For the interactive environment, a scheduler that uses measured speed-ups to adjust the processor allocation of each running job and thus improve response time is proposed. For batch environments, a scheduler that uses measured efficiencies to allocate processors in such a way as to maximise system efficiency is proposed. Both manual coded parallel applications and compiler parallelised sequential iterative applications were used in these experiments. Successive iterations of an application are reported as behaving similarly, so that measurements taken for a particular iteration are good predictors of near future behaviour. Thus, a measurement interval is equated to an application iteration, providing a basis by which to reasonably compare a job's performance as its processor allocation is varied. Several scheduling policies are specified and used to determine whether runtime measurements can be used by a scheduler to some gain. One of a number of main conclusions drawn was that despite the inherent inaccuracies of runtime measurements, and the added overhead of more frequent reallocations, schedulers using them can significantly outperform those that do not.

The Self Adjusting Scheduling (SAS) algorithm is introduced as being capable of improving the performance of programs by employing an on-line optimisation technique on a dedicated processor (Hamidzadeh and Lilja 1994). On the same theme of achieving high performance through efficient scheduling of independent tasks of a single program on the processors of a shared memory multiprocessor, a Self-Adjusting Dynamic Scheduling (SADS) algorithm is presented (Hamidzadeh and Lilja 1996). In each case, the scheduling strategy is described as centralised and employs a single processor of the multiprocessor to perform an on-line branch and bound technique that dynamically computes partial schedules based on the loads of other processors and the memory locality of the tasks and processors. SAS and SADS are described as overlapping the dynamic scheduling of tasks with the execution of the tasks themselves, thus no processor is idle while waiting for a task assignment. The technique performs partial task scheduling in repeated periods and assigns them to working processors' local queues until all tasks are scheduled. Each scheduling phase duration is determined by an on-line parameter tuning technique, which is self-adjusted based on working processor load. This self-adjustment of the time allocated to each scheduling period is aimed at minimising processor idle times. The particular task scheduling problem addressed is that of assigning the independent iterations of a parallel loop to processors in order to minimise the total execution time of the loop. The simulated performance results of the SAS algorithm show that the potential loss of performance caused by dedicating a processor to scheduling is outweighed by the higher performance produced by SAS's dynamically adjusted schedules. Experiments involving three SADS algorithms perform equally well with respect to the metrics of scheduling effort and scheduling quality.

Dynamic scheduling is defined as the management of computing resources according to their time-dependent states (Ahmad 1995). A need is identified for computing resource management techniques to efficiently perform dynamic task allocation, scheduling and load balancing. An ideal dynamic scheduling policy is described as one that has access to local and

global information such that it can cater for both small and large scale problems simultaneously. Common features of dynamic scheduling and dynamic load balancing algorithms are recognised as information policy, transfer policy and location policy.

In contrast to those discussed, Djordjevic and Tosic present a static scheduling technique which is described as having benefits over dynamic scheduling in that it is easier to realise and run-time overhead is eliminated (Djordjevic and Tosic 1996). A compile-time single-pass multiprocessor scheduling technique called Chaining is stated as finding a schedule for a task graph to be executed on a multiprocessor architecture such that execution time can be minimised. Chaining is stated as a graph transformation technique, which takes into account interprocessor communication, and takes n steps to transform a task graph into the scheduled task graph, where n is the number of tasks. This technique constructs the scheduled task graph incrementally by scheduling one task at each step. Two issues are considered when scheduling occurs, namely which task to schedule and where to place the scheduled task. The degree of parallelism is said to be limited by task precedence constraints. Parallelism dictates assignment of tasks to different processors, whereas communication overhead is removed when tasks are assigned to the same processor. Four chaining algorithms have been developed which, given a randomly generated task graph, perform approximately equally well. The quality of the simulation results is said to be dependent on the accuracy of communication time predictions, particularly when communication is medium or high.

2.3.4. Software Application Scheduling

The importance of efficiently scheduling distributed parallel applications in heterogeneous networks has been recognised (Berman et al. 1996). It is noted that since distributed resources can rarely be controlled by a single global scheduler, an application needs to be scheduled by the user. Both application and dynamic system information are stated as user considerations when scheduling so as to ensure greater performance. Fast networks are said to have made it possible to coordinate distributed resources for parallel applications that require more resources than are available at a single location. It is stressed that the drive behind the parallel implementation of much software is derived from the need to use collective memory systems rather than a desire for concurrent execution. An application specific approach to scheduling individual parallel applications on a heterogeneous system has been developed. Scheduling agents, termed Application Level Schedulers (AppLeS), are used to perform the scheduling activities of the user at machine speeds with more information. An AppLeS exists for each application and establishes performance efficient schedule, and implements the schedule with respect to the appropriate resource management system. Application and system specific models are used to predict application performance given a set resources. An AppLeS agent uses these models along with *resource load forecasts,* to select a resource set and an application schedule. A resource selection and scheduling algorithm are used to create schedules. The Network Weather Service (Wolski 1996) is used to provide dynamic system information. Predictive models are used to evaluate and rank candidate schedules. AppLeS is organised in terms of an active agent called the Coordinator, and four subsystems, namely the Resource Selector, the Planner, the Performance Estimator, and the Actuator. The Coordinator assesses the performance of the candidate schedules and selects a ''best'' schedule for implementation. The four subsystems extract information from a pool, which includes the Network Weather Service (Wolski 1996). The AppLeS approach is reported as being able to achieve substantial improvements for an individual application over traditional scheduling methods. AppLeS permits a user to deal with a heterogeneous system under the control of multiple system schedulers, shared by other contending applications, and able to a deliver only a dynamically varying fraction of resource performance.

A transactional workflow approach is presented, which provides a reliable execution environment for long running distributed applications (Ranno et al. 1997). The approach involves coordinating the execution of an application, which is modelled as a number of interdependent tasks. The implemented system ensures that tasks are scheduled for execution according to their dependencies, which may be temporal or dataflow. The system comprises of a number of components including a workflow execution service, which coordinates the execution of tasks that make up an application. Tasks are assigned controllers which can communicate with other controllers in order to ascertain when its associated task can commence. Most of the service components are provided through Common Object Request Broker Architecture (CORBA) interfaces which enables interoperability with other systems (OMG 1997).

2.3.5. Scheduling and Load Balancing

Clement and Durfee focus on the task scheduling and load balancing issues of a multi-agent system

(Clement and Durfee 1998). Tactical Assistants for Interaction Planning and Execution (TAIPE) is named as a multi-agent system used by the United States Navy to assist in automating ship operation. The need to balance task assignment loads and schedules of human operators is recognised. A feature of TAIPE is that task arrival rate is relatively low and task processing time is relatively high, thus motivating near optimal distributed scheduling. A heuristic hill-climbing algorithm has been found to be suitable for managing task loads and schedules in TAIPE. Task context switching is a feature of the multi-agent system that is taken into account. Clustering is used to group similar tasks and thus minimise context switching. Two load balancing strategies are supported by the hill-climbing algorithm. The first allows the operator to locally search for a good cluster of tasks. The second is a centralised search through task reallocations among two or more operators to improve the coherence of clusters at each operator and to balance load.

The performance of a computing environment consisting of workstations, a high-speed interprocessor communication network, and shared resources, can suffer due to load imbalance (Svensson 1990). One cause of load imbalance is identified as the inappropriate remote execution of certain jobs. A filter component, called History, within a load sharing algorithm is used to identify those jobs not suitable for remote execution. Previous execution statistics are used to define a filter threshold to ensure that short lived jobs are executed locally. Jobs not considered to be short lived are passed to the load balancing algorithm which are then executed remotely in order to improve performance.

Load balancing is used by task assignment algorithms to achieve optimal performance across a system of hosts (Crovella et al. 1997). This is stated as not true for distributions characterised by the high majority of tasks being very small and in excess of half the load comprising of a low minority of large tasks, which are known as heavy-tailed task size distributions. Size Interval Task Assignment with Variable Load (SITA-V) is introduced as a task assignment policy that places small tasks on lighter loaded hosts while large tasks are assigned to heavily loaded hosts. Task slowdown, rather than task waiting time, is considered as the important performance measure in a heavy-tailed task size distribution. Thus, SITA-V completes small tasks more quickly while taking longer to complete long tasks. Improving slowdown does not alter system utilisation but does increase task waiting time. Therefore, SITA-V reduces slowdown by unbalancing the load. This is in direct contrast to the concept of balancing load to reduce waiting time.

A general set of criteria is proposed to facilitate the review and comparison a number of cluster management and queuing systems (Kaplan and Nelson 1994). These systems include Codine, Condor, DQS, Load Balancer, LoadLeveler, LSF, and NQE. Twenty-four evaluation criterion are identified to asses whether the systems supported features such as heterogeneity, parallelism, message passing, checkpointing, migration, load balancing, single point failure, and network file system. In addition, a question criteria is applied to each system being reviewed. The evaluation of the systems gave rise to several important conclusions. The choice of a suitable system is dependent on the needs of the user. Cluster management and queuing systems are continually advancing technologically, thus necessitating an up to date review of such packages by systems administrators if their use is being considered.

Antonioletti conducts a more recent, and almost identical, review of six of the twelve systems, termed load sharing packages, originally surveyed by Kaplan and Nelson (Antonioletti 1997). The concepts of load sharing is also discussed including the role of scheduling. Scheduling is described as needing policies regarding load balancing, load assessment, and placement. Load balancing policies are stated as either dynamic or static. Load assessment policies are described as global, central or distributed. Placement policies named include random, threshold, reserve and lowest.

3. Conclusion

Coordination has been identified as a concept that can potentially lead to an improvement in the design process and consequently product success. Since the late 1980s and early 1990s, coordination has started to receive the attention of many researchers from a variety of disciplines including engineering design, computer science, and organisation theory. Consequently, a wide range of approaches, techniques, systems, algorithms, models, tools and procedures have been developed in an attempt to address the concept of coordination. Despite this growing amount of research, and the respective approaches and/or systems developed, there remains the lack of a single widely accepted perception of coordination and its constituents. While an increasing amount of research is being performed, currently there appears to be relatively little research being undertaken which attempts to satisfy the requirement to comprehensively under-

stand the broader aspects of coordination. Indeed, even within the same discipline, there is a clear difference in opinion as to what constitutes coordination. This has resulted in a rather liberal use and application of the word 'coordination', and thus a diversity in definitions and understanding of the concept.

The papers studied for the purpose of this survey indicate that while the issue of coordination exists within a number of disciplines, there are areas of commonality between each. Task management, scheduling and planning, and resource management are those issues which appear to be most prominent with respect to operational coordination. However, no single approach has yet been developed to encapsulate each of these issues and the relationships between them such that an agreed approach can be established.

Three main recommendations arise from this review with respect to coordination. First, research in academia, along with the support of industry and the case studies they can provide, should continue to be supported in the exploration for a unified understanding and approach to coordination. Secondly, a clear need exists to bring together those academics, researchers and industrialists with an interest in coordination with the intent of identifying all of the fundamental issues involved. Finally, there is a need for further collaboration between academics, researchers and industrialists from different disciplines to pool ideas and theories pertaining to coordination. Comparisons and contrasts between the comprehension and perception of coordination from the viewpoint of academics, researchers and industrialists from the various discipline's will enable an even greater understanding. Only when these key issues have been addressed can there be a collective and unified understanding of coordination.

Acknowledgements

The authors gratefully acknowledge the support given by the Engineering and Physical Science Research Council who provided the grant RES/4741/0920 that enabled this work to be undertaken.

References

Ahmad I (1995) Editorial: Resource management in parallel and distributed systems with dynamic scheduling: dynamic scheduling. Concurrency: Practice and Experience 7(7):587–590

Aikat S (1996) Using PERT to plan and schedule your documentation projects. Society for Technical Communication Annual Conference, pp 71–74

Altus SS, Kroo IM, Gage PJ (1996) A genetic algorithm for scheduling and decomposition of multidisciplinary design problems. Trans ASME 118:486–489

Andreasen MM, Duffy AHB, MacCallum, KJ et al. (1996) The Design Co-ordination Framework: key elements for effective product development. International Engineering Design Debate: The design productivity debate, pp 151–174

Antonioletti M (1997) Load Sharing Across Networked Computers. Edinburgh Parallel Computing Centre, Version 1.0

Bendeck F, Goldmann S, Holz H et al. (1998) Coordinating management activities in distributed software development projects. IEEE Post-Proceedings of the 7th International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises

Berman F, Wolski R, Figueira S et al. (1996) Application-level scheduling on distributed heterogeneous networks. Proceedings of Supercomputing '96

Bullinger H-J, Warschat J (1996) Concurrent Simultaneous Engineering Systems – The way to successful development, Springer-Verlag, London

Carter IM, MacCallum KJ (1991) A software architecture for design co-ordination. Artificial Intelligence in Design Conference, pp 859–881

Chan FTS (1997) Resource management in project scheduling through simulation, Int J Computer Applications in Technology 10(1/2):81–89

Cho JG, Yum BJ (1997) An uncertainty importance measure of activities in PERT networks. Int J Prod Res 35(10):2737–2757

Clement BJ, Durfee EH (1998) Scheduling High-Level Tasks Among Cooperative Agents. Proceedings of International Conference on Multi-Agent Systems '98

Coates G, Duffy AHB, Whitfield RI et al. (1999a) A methodology for design coordination in a distributed computing environment. Proceedings of the 12th International Conference on Engineering Design, Munich, Germany

Coates G, Duffy AHB, Hills W et al. (1999b) Enabling concurrent engineering through design coordination. 6th ISPE International Conference on Concurrent Engineering, Bath, UK

Crovella ME, Harchol-Balter M, Murta CD (1997) Task Assignment in a Distributed System: Improving Performance by Unbalancing Load. Boston University, Technical Report BUCS-TR-1997-018

Crowston K (1996) A Taxonomy of Organizational Dependencies and Coordination Mechanisms. Technical Report 174, Massachusetts Institute of Technology, Centre for Coordination Science

Davis MB, Sydir JJ (1996) Position paper: Resource management for complex distributed systems. Proceedings of the Workshop on Object-Oriented Real-Time Dependable Systems (WORDS), pp 113–115

Decker KS, Lesser VR (1995) Coordination Assistance for Mixed Human and Computational Agent Systems. UMass Computer Science Technical Report 95–31

Dellen B, Maurer F (1996) Integrating planning and execution in software development processes. Proceedings of WETICE-96, Stanford, CA

Di Battista G, Pietrosanti E, Tamassia R et al. (1989) Automatic layout of PERT diagrams with X-PERT. IEEE Workshop on Visual Languages, pp 171–176

Djordjevic GL, Tosic MB (1996) A compile time scheduling heuristic for multiprocessor architectures. The Computer J 39(8):663–674

Duffy AHB, Andreasen MM, MacCallum KJ et al. (1993) Design coordination for concurrent engineering. J Eng Design 4(4):251–265

Durfee EH (1993) Organisations, plans, and schedules: An interdisciplinary perspective on coordinating AI systems. J Intelligent Systems, Special Issue on the Social Context of Intelligent Systems, 3(2–4)

Durfee EH, So YP (1997) The effects of runtime coordination strategies within static organizations. Proceedings 15th International Joint Conference on Artificial Intelligence (IJCAI97)

Ephrati E, Rosenschein JS (1994) Divide and Conquer in Multi-agent Planning. AAAI, 1:375–380

Findler NV, Elder GD (1995) Multiagent coordination and cooperation in a distributed dynamic environment with limited resources. Artificial Intelligence in Eng 9:229–238

Garvey A, Humphrey M, Lesser V (1993) Task interdependencies in design-to-time real-time scheduling. Proceedings of the 11th National Conference on Artificial Intelligence

Garvey A, Lesser V (1994) Design-to-time real-time scheduling. IEEE Trans Syst, Manage and Cybernetics, Special Issue on Planning, Scheduling and Control, 23(6)

Greenwood RM (1995) Coordination theory and software process technology. In: W.Schafer (ed), Software Process Technology: Proceedings of the 4th. European Workshop (EWSPT'95), Vol. 913 of LNCS, Springer-Verlag, Noordwijkerhout, The Netherlands

Greenwood RM, Kawalek P, Robertson I et al. (1997) Modern Systems Architecture: The Contribution of the Coordination Layer. Workshop Research Directions in Process Technology, France

Goldmann S (1996) Procura: A Project Management Model of Concurrent Planning and Design. Proceedings of WETICE '96, Stanford, CA

Hamidzadeh B, Lilja DJ (1994) Self-adjusting scheduling: An on-line optimization technique for locality management and load balancing. International Conference on Parallel Processing, 11:39–46

Hamidzadeh B, Lilja DJ (1996) Dynamic scheduling strategies for shared-memory multiprocessors. International Conference on Distributed Computing Systems, pp 208–215

Huber MJ, Durfee EH (1995) On acting together: Without communication. AAAI Spring Symposium on Representing Mental States and Mechanisms, Stanford, CA, AAAI Press, pp 60–71

Jennings NR, Jackson AJ (1995) Agent-based meeting scheduling: A design and implementation. IEE Electronics Letters 31 (5) 350–352

de Jong E (1997) Multi-agent coordination by communication of evaluations. Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '97

Kamburowski J (1985) Normally distributed activity durations in PERT networks. Journal of Operational Research Society 36(11):1051–1057

Kaplan JA, Nelson ML (1994) A Comparison of Queueing, Cluster and Distributed Computing Systems. NASA Langley research Center, NASA TM 109025

Kim J, Lilja DJ (1998) A Network Status Predictor to Support Dynamic Scheduling in Network-Based Computing Systems. Technical Report No. HPPC-98-11, High-Performance Parallel Computing Research Group, University of Minnesota

Liu JS, Sycara KP (1996) Multiagent coordination in tightly coupled task scheduling. ICMAS Proceedings, Kyoto, Japan, AAAI Press

Malone TW, Crowston K (1994) The Interdisciplinary Study of Coordination. ACM Comput Surv 26(1):87–119

Maurer F (1996) Project coordination in design processes. Proceedings of WETICE '96, IEEE Press

Moreira JE, Naik VK (1997) Dynamic resource management on distributed systems using reconfigurable applications. IBM J Res and Development 41(3):303–330

Nguyen TD, Vaswani R, Zahorjan J (1996a) Maximising speedup through self-tuning of processor allocation. Proceedings of the 10th International Parallel Processing Symposium

Nguyen TD, Vaswani R, Zahorjan J (1996b) Using runtime measured workload characteristics in parallel processor scheduling. In: DG Feitelson, L Rudolph (eds), Job Scheduling Strategies for Parallel Processing, Vol. 1162 of Lecture Notes in Computing Science, Springer-Verlag, Berlin

Object Management Group (1997) The Common Object Broker: Architecture and Specification: Revision 2.1. OMG

Ord JK (1991) A simple approximation to the completion time distribution for a PERT network. J Operational Res Soc 42(11):1011–1017

Quinn JF (1994) Integrating UNIX systems into a resource management repository. Proceedings of the 20th International Conference for the Resource Management & Performance Evaluation of Enterprise Computing Systems, 2:812–822

Ranganathan M, Acharya A, Salt J (1996) Distributed resource monitors for mobile objects. Proceedings of IWOOOS '96

Ranno F, Shrivastava SK, Wheater SM (1997) A system for specifying and coordinating the execution of reliable distributed applications. International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97), Cottbus, Germany

Smith RG (1980) The Contract Net Protocol: High-level communication and control in a distributed problem solver. IEEE Trans Computers C-29(12)

Soroush HM (1994) The most critical path in a PERT network. J Operational Res Soc 45(3):287–300

Svensson A (1990) History, an intelligent load sharing filter. Proceedings of the 10th International Conference on Distributed Computing Systems, pp 546–552

Tan GW, Hayes CC, Shaw M (1996) An intelligent-agent framework for concurrent product design and planning. IEEE Trans Eng Manage 43(3):297–306

Todd DS (1997) Multiple Criteria Genetic Algorithms in Engineering Design and Operation. PhD Thesis, University of Newcastle upon Tyne

Whitfield RI, Coates G, Duffy AHB et al. (2000) A review of coordination approaches and systems – Part I: A strategic perspective. Res Eng Design 12:48–60

Wolski R (1997) Forecasting network performance to support dynamic scheduling using the Network Weather Service. International Symposium on High Performance Distributed Computing, pp 316–325

Yen IL, Bastani FB (1995) Robust parallel resource management in shared memory multiprocessor systems. IEEE Symposium on Parallel & Distributed Processing, pp 458–465

Zhu Z, Heady RB (1994) A simplified method of evaluating PERT/CPM network parameters. IEEE Trans Eng Manage 41(4):426–430

Appendix A

Table 1. Coordination systems – summary of features

System	Type of coordination	Techniques used	Application
AGENDA	Planning, decomposition, and scheduling	Genetic algorithm	Multi-disciplinary optimisation
AppLeS	Application specific approach to scheduling	Several agents along with NWS enable efficient scheduling	Individual parallel application
COFCAST	Look-ahead coordination	A blackboard model with agents communicating information	Job-shop schedule optimisation
CoMo-Kit	Planning and execution	A Modeler creates plans and a Scheduler provides planning and execution support	Software development
DCS	Task management, resource management and monitoring, scheduling and planning, and task execution	Agents work in a cooperative manner to efficiently perform the design analysis	Distributed computational design analysis
DRMS	Resource management	Functional components enable resource control and scheduling	Reconfigurable parallel applications
HOBS	Organisation and control, integration and modelling	Blackboard system comprising of an Executive control mechanism and knowledge sources	Generator design
Komodo	Resource monitoring	Software daemons ensure variations in network utilisation are considered	Mobile programs
MSA	Scheduling	Agents act on behalf of users to schedule mutually convenient meetings	Not indicated
NSP	Resource monitoring	A distributed system that detects and predicts network load	Support resource scheduling
NWS	Forecasting	Distributed sensors obtain network information used to forecast future conditions	Parallel application scheduling
PGP	Unites organisational theory, planning and scheduling	Agents are coordinated to cooperatively solve problems	Distributed vehicle monitoring
Procura	Planning, scheduling and execution	Agents interleave planning, scheduling and execution of design projects	Project management
TAIPE	Scheduling and load balancing	Multi-agent system to balance task loads and scheduling of humans	Automating ship operation