Coordination Approaches and Systems – Part I: A Strategic Perspective

Robert Ian Whitfield[1], Graham Coates[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, Department of Design Manufacture and Engineering Management, University of Strathclyde, Strathclyde, UK

Abstract. *This is the first part of a two-part paper presenting a fundamental review and summary of research of design coordination and cooperation technologies. The theme of this review is aimed at the research conducted within the decision management aspect of design coordination. The focus is therefore on the strategies involved in making decisions and how these strategies are used to satisfy design requirements. The paper reviews research within collaborative and co-ordinated design, project and workflow management, and, task and organization models. The research reviewed has attempted to identify fundamental coordination mechanisms from different domains, however it is concluded that domain independent mechanisms need to be augmented with domain specific mechanisms to facilitate coordination. Part II is a review of design coordination from an operational perspective.*

Keywords: Cooperation; Design coordination; Organisation models; Project management

1. Introduction

This review represents one part of a two-part paper, the combined aim of which is to summarize the current state-of-the-art of coordination techniques and methodologies. The focus of this paper is on coordination from a strategic perspective, which may be viewed as the management of the control mechanisms that govern a particular process. The paper is not an exhaustive review of all of the literature available, but is intended to give the reader an understanding of the broad spectrum of work that has been undertaken within this field.

A great deal of the research investigated here has similarities between how humans cooperate and organize their communities and manage their problems. Autonomous agents are increasingly being applied to solve the problems associated with the distributed nature of design activity. The agents have been developed to possess knowledge of their domain and be capable of reasoning about the tasks and activities for which they hold responsibility.

Increases in computational power have meant that not only is an increasing amount of design and other activity being performed within agent architectures, but the control mechanisms that once governed how a design artefact progressed through its lifecycle are also being automated. The automation of this control has been investigated using a number of different approaches, however, one conclusion is usually drawn; that the control, like the agents, should be distributed. Distributing the control mechanisms in this manner has been of fundamental importance when producing communities of agents, cooperating together to solve a particular problem. However, the authors believe that cooperation needs to be augmented with rationale to achieve coordinated behaviour within a multi-agent design system, i.e. agents may be able to work cooperatively to solve a particular problem, but the agents must be able to reason, rationally, if the design problem is going to progress, otherwise chaos may occur (Hogg and Huberman 1991). It is also believed that an outcome of providing coordination would be to enable the identification of design activity which may be undertaken concurrently (Duffy et al. 1993).

1.1. Scope

This paper is intended to give the reader an idea of the research that has been, and is currently being undertaken within the field of design coordination. Consequently, the review is an interpretation of a wide range of published research. Naturally, the papers presented here have been summarized to what is believed to be appropriate and may therefore not truly reflect the authors' original intentions. The reader is referred to the papers for a more complete coverage.

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

The work covered within this review is generally based around the development of coordination mechanisms for computer-based Distributed Artificial Intelligence (DAI) systems. This field has drawn on fundamental aspects of coordination such as communication, negotiation and the use of knowledge from practical applications, and formalized them within a computer-based domain. It is believed that this approach will consequently help to re-define how coordination is managed within actual applications.

1.2. Organisation

The methodologies reviewed within this paper are organized within six broad subject areas in an attempt to distinguish between the different approaches adopted. The approaches identified are:

- coordination techniques,
- cooperation techniques,
- organization models,
- project management,
- workflow management, and
- task models.

In general the methodologies reviewed tend not to fit neatly into these categories, and it may be argued that some of these categories may be subsumed within others. The methodologies have however been categorized based upon the general theme of the work described.

2. Design Coordination Methodologies

Design coordination is a subject that is not easily defined. This is apparent from the amount of literature that attempts to solve coordination problems using different perceptions of what coordination actually is. The methodologies tend, however, to have an agreed starting point, this being the use of autonomous agents having roles and responsibilities, working within a community or organization. The use of agents within this particular manner illustrates the use of cooperation techniques to enable a particular problem to be solved. Coordination techniques may be used within this type of framework to enable an organization of cooperative agents to solve a problem in an organised and efficient manner. Two approaches have been adopted to enable coordinated activity to be undertaken, either using a formal representation of the individual tasks required or by representing the entire process using either project or workflow management.

2.1. Coordination Techniques

The techniques and methodologies described here vary considerably from coordination testbeds that are used to represent behavioural aspects, to languages that allow agents to discuss coordination issues, and algorithms that describe procedures for performing activity within a coordinated manner. What is apparent is that there is a great deal of diversity in the approaches that have been devised to tackle coordination each of which is able to represent knowledge of the coordination process.

A flexible testbed called MICE (Michigan Intelligent Coordination Experiment) was developed to simplify the investigation and evaluation of coordination mechanisms (Durfee and Montgomery 1989). The authors refer to previous investigations (Smith 1980; Lesser and Corkill 1983), that demonstrated coordination techniques applied within domain specific examples and argued that the use of this coordination knowledge was difficult to utilize within other domains. Their investigation was conducted by simulating an agent environment through the encoding of domain specific environmental constraints and characteristics that influenced the coordination process. It was intended that this method may be used to extract general principles and techniques for coordination with the behaviour of the simulation highlighting any discrepancies. Durfee demonstrated the use of MICE by constructing environments that simulated predators chasing prey, agents fighting a fire, and robots working together. A summary of MICE, and the other systems that are reviewed within this paper can be seen within Table 1.

A coordination language for multi-agent systems called COOL was developed around the notion of structured conversation between agents allowing knowledge to be captured, represented and used by agents to interact sociably (Barbuceanu 1995; Barbuceanu and Fox 1996b). As with other multi-agent communities (Cutkosky et al. 1993; Lesser 1998), the agents within Barbuceanu's framework did not have a complete representation of the environment in which they operated. The language however enabled coordination to be managed allowing the agent to communicate its requirements and actions to other agents.

The COOL coordination language was used within an agent building shell to provide communication, knowledge management, cooperative information distribution, organization modelling and conflict management (Barbuceanu and Fox 1996a). The agent building shell enabled coordinated, distributed

systems to be developed from interoperable and reusable building blocks.

Barbuceanu and Fox recognised the need for the agents to possess not only domain specific knowledge to enable them to undertake their activities, but also for knowledge of how the agents should interact with each other. This social knowledge has been used to define an agent's role within the agent community and has subsequently been used to define organizations of agents (Durfee 1993; So and Durfee 1996). Jennings also defined the term 'cooperation knowledge level' as the social interaction which describes how agents should cooperate to solve a common problem (Jennings 1992).

The Partial Global Planning (PGP) mechanism falls into a discipline called Cooperative Distributed Problem Solving (CDPS) which develops ideas from artificial intelligence and distributed computing (Durfee and Lesser 1991). CDPS is the study of how multiple intelligent systems collectively solve problems which were beyond their individual capabilities. Starting with local coordination information, agents responded to their current situation, exchanging information with other agents to form a non-local or partially global plan. The authors hypothesized that coordination is not a separate phase in group activity, but is instead an integral part of decision making. They added that coordination arises out of local planning rather than by providing a protocol or language to allow systems to communicate. The PGP mechanism was demonstrated within the modelling of the activity of a distributed vehicle monitoring testbed.

The iDCSS system attempted to combine existing coordination approaches to avoid many of their individual limitations (Klein 1995). Three types of coordination technology were reviewed; process management, conflict management, and rationale capture and the advantages and disadvantages of each type of method were highlighted. Each of these technologies focused on handling one particular type of dependency and was subsequently found to have difficulties in handling others. The iDCSS system was an attempt to 'combine the strengths and avoid the weaknesses' of each approach to enable the coordination of cooperative design.

The Generalised Partial Global Planning (GPGP) approach was an extendable set of algorithms that enabled cooperative computational agents to determine the importance of certain tasks and schedule their activities based upon the most appropriate time (Decker and Lesser 1992; Lesser et al. 1998). The algorithms were produced by examining and generalising the mechanisms developed for the PGP algorithm. The focus of the work was more on detecting the coordination relationships than on local scheduling (Decker and Lesser 1995). The authors claimed that the algorithms could be applied to any computational domain that required coordination, but were again tested with a re-implementation of the vehicle monitoring testbed (Durfee and Lesser 1991). The algorithms were defined using TÆMS (Task Analysis, Environment Modelling, and Simulation), which was described as a framework for expressing coordination problems by representing an agent's current beliefs about the structure of the tasks in the current episode (Decker 1996). Five mechanisms were produced to respond to certain features of the agent's current task environment and were intended to work in conjunction with the agent's local scheduling mechanism to effectively produce improved local schedules.

The GPGP framework was compared with a centralized scheduling agent measuring the performance of the total final quality achieved by the system, the amount of work done, the number of deadlines missed and the termination time. The authors discovered that the GPGP system was outperformed by the centralized system 57% of the time whilst producing 85% of the quality. This was accredited to the additional time taken for the GPGP based agents to gather a partially global plan of the design problem, compared with the centralized system's already existing complete view. The use of a distributed system was preferred however by the possibility of failure of the centralised scheduling agent resulting in complete failure of the system. The authors also add that increasing numbers of agents and task structures would make a complete view of the episode hard to achieve using the centralized approach. Finally, the authors acknowledge that the five mechanisms only dealt with certain aspects of coordination and may be extended as new ideas arise.

The TÆMS and GPGP paradigms were the foundations for a domain-independent agent architecture that assumed the existence of a local scheduling mechanism and a level of self-awareness regarding the agent's current and intended goals (Lesser 1998). The agent architecture was capable of solving problems subject to real-time constraints, the availability of resources and the ability to coordinate with the problem-solving activities of other agents. Lesser's agents provided multi-agent scheduling, organizational design, detection and diagnosis and on- line learning. Providing such mechanisms enabled the agents to execute their short term coordination behaviors as well as altering their role within an organization.

The Design Coordination Framework proposed a

concept for an 'ideal' design coordination system (Andreasen et al. 1996). The authors identified the recent trend of moving away from isolated design towards the concurrent implementation of the design process. Concurrent Engineering is becoming increasingly possible resulting from the development of computer based tools and integrated product data models, however, the authors suggest:

> ''A major shortcoming of the Concurrent Engineering view is the failure to recognize that what is truly required is not for activities to be carried out in parallel but for resources to be effectively utilized in order to carry out tasks for the right reasons, at the right time, to meet the right requirements and give the right results. That is: the key to achieving optimal design performance, and hence design productivity, is the effective coordination of the design process.''

To this end, Andreasen identified a number of frames representing the key elements of coordination which reflect the states of the design process.

Jennings argued that commitments (pledges to undertake a specific course of action) and conventions (means of monitoring commitments in changing circumstances) were the foundation of coordination in all DAI systems (Jennings 1996). The objectives of the coordination process were proposed as ensuring:

> ''that all necessary portions of the overall problem are included in the activities of at least one agent, that agents interact in a manner which permits their activities to be developed and integrated into an overall solution, that team members act in a purposeful and consistent manner, and that all of these objectives are achievable within the available computational and resource limitations.''

Jennings discussed the coordination of the actions of a group of agents based upon three different scenarios. The first scenario involved all of the agents having a complete representation of the goals, actions and interactions associated with all of the other agents within the community. If each agent was subsequently given an unlimited amount of processing power, Jennings argued that the agents would know the activities of each of the other agents, and would be able to predict what they would be doing in the future. This scenario was considered to be impracticable due to the increase in computational power and communication required to maintain an agent's representation being considerably greater than that available to perform the activity.

The next scenario was to furnish a single agent with a global representation and use this agent to direct the activity of the other agents. This scenario would significantly reduce the resource requirements of the individual agents but would result with the global controller having a communication bottleneck as well as the possibility of suffering from complete failure.

The final scenario involved distributing the control and data such that each agent has a degree of autonomy in deciding the activity that it is going to undertake. This scenario is perhaps more representative of the control mechanisms involved within actual organizations. One of the requirements of this scenario is that the design of the control mechanisms should not significantly restrict the resources available to perform the activity – a condition that Decker and Lesser only partially satisfied (Decker and Lesser 1992). Each agent also has only a partial perspective of the global problem.

Jennings finally hypothesized that all coordination mechanisms could be expressed in terms of commitments, conventions, social conventions and local reasoning capabilities, which was demonstrated by the reformulation of a number of common coordination techniques using these terms.

Gupta et al. (1996) described a constraint based system with the objective of achieving high levels of concurrency by timely communication and coordination of information. The authors discussed two approaches that have been traditionally used to obtain concurrency; multi-disciplinary team meetings, and distribution lists. They mentioned that team meetings are often difficult to arrange with respect to deciding who needs to attend, wasteful of time, and are frequently complicated by geographical distribution. The dynamic nature of organizations also highlights limitations of distribution lists due to their static representation. The timely communication of accurate information, the ability to share work in progress, and the accurate determination of the impacts of change were all requirements that were identified to achieve concurrency.

The framework that was produced possessed a representation of requirements, constraints, parameters, roles and responsibilities within a shared object model. Gupta implemented this model of concurrency by providing an environment in which people could share their work and work on the same piece of data.

2.2. Cooperation and Collaboration Techniques

Smith specified speed, reliability, extensibility, and the ability to handle applications that have a natural spatial distribution as reasons for having a distributed approach to problem solving (Smith 1980). Reducing bottlenecks was one area where speed may be achieved by distributing control, data and knowledge sources. This philosophy has been adopted by a number of significant research projects since and is still seen as being one of the ways forward within the artificial intelligence community. Distribution of control also enhances reliability and permits graceful degradation of performance in the case of individual agent failures.

One of the earliest research projects investigating cooperative problem solving was the contract net protocol which described problem solving communication and control for the agents within a distributed system (Smith 1980). The agents were defined as loosely coupled, asynchronous knowledge sources. Smith presented the connection problem which identified the issues associated with how one agent with a task to be performed may negotiate with another, suitably idle agent, to undertake this task. Four important components were recognised to facilitate such negotiation; local rather than centralized process control, a two way exchange of information, evaluation of information from each agent's perspective, and final agreement achieved by mutual selection. The contract net protocol has since fashioned many other researchers ideas within the area of negotiation.

Smith has argued that solutions to problems within an artificial intelligence environment often require many tasks to be performed and the combination of many tasks with many knowledge sources can result with a combinatorial explosion in the amount of activity.

In a discussion about the trends in CDPS, an attempt was made to justify why the difficulties of defining an effective cooperation mechanism should be overcome and consequently warrant designing distributed problem solving systems (Durfee et al. 1989). The authors recognized that advances in computational power and communication systems make the connection of large numbers of powerful processors a cost-effective way of dealing with the computational requirements of DAI. They add that artificial intelligence systems are invariably distributed and maintenance and debugging of smaller packets of distributed knowledge is therefore an easier task than is the case for a single unifying system. Providing distribution of knowledge also enables the development and understanding of the mechanisms involved within cooperation and coordination.

From their analysis of a number of approaches for CDPS, the authors identified three requirements for effective coordination. The first requirement was for a structured approach to enable the agents to interact in a predictable way. The dynamic nature of the environment in which CDPS agents operate requires that the agents are flexible in dealing with incomplete, inaccurate or obsolete information. The final requirement was that the agents should possess the knowledge and reasoning capabilities to be able to intelligently use this structure and flexibility.

The Design Fusion system was intended to produce superior designs by the concurrent consideration of multi- disciplinary design requirements and used constraints as the language with which the designers communicated with each other (Finger et al. 1992). The designers were given advice regarding the analysis of the design via groups of perspectives which contained domain specific knowledge sources. Coordination of the perspectives was managed through a centralized blackboard architecture.

Cooperative multi-agent systems could be constructed using the GRATE framework which contained a significant amount of cooperation and control related knowledge that could be augmented with domain specific knowledge (Jennings et al. 1992). The authors recognised that the ideal scenario would be for the framework to have all of the necessary control knowledge to enable cooperation between any suite of agents. They realised however, that it would be unlikely that the cooperation mechanism would be expressive enough to be able to deal with the specifics of the agent's domain. Despite this, they believed that generic control knowledge should be an integral part of the solution but recognized that it needed to include more powerful mechanisms.

Explicitly represented knowledge was used as a communication medium to facilitate cooperation within the SHADE project (Gruber et al. 1992). The aim of the SHADE project was to provide a framework that would enable the sharing of knowledge between people and their programs and to coordinate that knowledge so that it reached the interested parties. The knowledge-based technologies produced by DARPA were used as the basis for the development of this framework.

A hybrid approach using control knowledge and more powerful reactive mechanisms was developed within the ARCHON project (Jennings and Pople 1993; Wittig et al. 1994). Jennings proposed that the control mechanisms within a distributed system

should also be distributed to overcome the difficulties in representing the control strategies of legacy systems within a unifying whole and the associated bottlenecks that such a system would produce.

The initial objective of the ARCHON project was to provide a supporting software architecture to enable the cooperation of legacy expert systems dealing with different aspects of decision making. Jennings addressed a number of issues that are key considerations within the development of multi-agent systems. ARCHON was intended to be used within industrial applications, hence it was identified that the decision making process should not consume significant amounts of resources and leave sufficient computational power for the design activity. The distribution of the decision making process would therefore become a function of the distribution of available resources ensuring that all events are dealt with giving priority to those of greater importance. The agents were given roles, either to complete their own objectives, to aid in the completion of other agents' objectives, or as a mixture of the two. Jennings claimed that from the perspective of DAI, ARCHON represented; ''one of first serious attempts to build a generic cooperation framework for large-scale, real-world industrial applications''. An example was given of the use of the ARCHON framework within the fault diagnosis in an electricity management application (Wittig et al. 1994).

The Palo Alto Collaborative Testbed (PACT) was formed to explore the issues associated with building a multi-disciplinary framework that primarily dealt with the sharing of knowledge between disparate computer-aided design software (Cutkosky et al. 1993). The authors examined the technological and sociological issues of building large scale, distributed concurrent-engineering systems. One of the problems identified was that existing legacy software generally tended to offer 'point' solutions to problems within their particular domain, and whilst the tool may contain its own knowledge-base and representation of the design model, the methods available to transfer this knowledge to other design tools were either non-existent or very limited.

The group realized that the answer lay not in unifying the systems, but in providing an over-arching framework that could coordinate the tools without having to change them. To achieve this, the framework communicated the tool's knowledge in a formal manner using techniques developed by the DARPA sponsored knowledge-representation standards committees (Patil et al. 1992). The resulting agent based architecture was demonstrated using examples including cooperative design refinement, simulation of the design artefact, and the interactions associated with a design change.

Toye et al. (1994) examined the issues associated with providing a shared perception of the design and the design process which was not previously possible due to the lack of information organization within the available CAD systems. This deficiency resulted in difficulties involving the retrieval of past designs or the lack of information available within these designs. The SHARE system was designed to identify what was meant by a 'shared understanding', and to identify the tools that would enable designers to capture and exploit it and produce design records which may be shared and retrieved for future use.

A similar problem was described and tackled within a distributed collaborative design methodology using the Internet called Madefast (Cutkosky et al. 1996). The Madefast project was an exercise in using Internet-based tools developed by the ARPA Manufacturing Automation and Design Engineering program to produce a shared repository of all of the models, notes, results and calculations relating to the design of an artefact. The team realized that it was as important for the tools to capture the processes and rationale leading to the design as well as the design itself.

2.3. Organisation Models

Despite not specifically describing the development of an organization model, Jennings described the social interaction knowledge required for the behavior of agents collaborating to solve a common problem (Jennings 1992). This 'cooperation knowledge level' was developed to provide agents with rich and explicit models of common social phenomena. Early distributed systems commonly only contained the knowledge required to perform their domain specific activities, and subsequently had difficulty in cooperating to solve a common task. To solve this problem, models of knowledge describing how agents should perform sociably to solve a common problem were developed.

One of the foundations for the development of DAI has been the analysis of how communities of people collectively solve problems, the intention therefore being to define how AI agents may be individually and collectively intelligent. Durfee focussed on this to determine three social metaphors for the basis of DAI; organizations, team plans and collective scheduling (Durfee 1993). From an organizational perspective, each AI agent would play a particular organizational

role. If these roles were defined and assigned appropriately, the group as a whole could work as a coherent team.

The CommonKADS framework was developed to enable the acquisition of knowledge within cooperating knowledge-based systems (Weih et al. 1994). The CommonKADS model set was described by a set of models representing different types of knowledge. An organizational model (de Hoog et al. 1994a) contained descriptions of the function, structure, authority and resources of an organization. The framework also contained models of tasks, agents, communication and expertise (de Hoog et al. 1994b).

The analysis of human organizations has contributed to the development of the field of distributed artificial intelligence. So and Durfee described the design of tree-structured organizations consisting of communicating, autonomous computational agents (So and Durfee 1996), and defined an organization as ''a long term commitment made by the agents to a particular way of jointly handling the cooperative tasks''.

Previous distributed systems performed their activity based upon a set of instructions. So and Durfee mention that as computational power increases, it is possible to give the agents a degree of ''intelligence'' by providing them with roles and responsibilities. The paper goes further by demonstrating how the agents can themselves design the organization such that they can participate in jointly performing a set of tasks in a cooperative manner.

The issues regarding organizational self-design (So and Durfee 1994) were discussed following the concern that organizational structures are typically dynamic in nature, and as such, the agents operating within an organization need to be able to adapt their role and responsibilities so that they can adjust as the circumstances change.

An organization ontology for the TOVE (Toronto Virtual Enterprise) model was described which possesses a number of agreed upon organization related concepts such as agents, roles, positions, goals, communication, authority and commitment (Fox et al. 1996). An organization was defined as a set of constraints on the activities performed by a set of collaborating agents. The authors believed that the use of an ontology made it possible for both mixed groups of agents and people to agree upon the requirements by ensuring that everybody has the same interpretation. They described the organization ontology as being composed of a number of agents playing roles in which they are acting in solving specific goals according to various constraints defining the 'rules of the game'.

Barbuceanu also used an ontology to describe the organizational structure within which the agents were operating (Barbuceanu 1996a). As with Fox et al. (1996), the ontology consisted of models of the other agents within the environment, their roles, goals and actions available to them, the information that they were interested in and the services that they could provide.

Nagendra Prasad et al. (1996) described a graph-grammar-based task structure language to enable the design of functionally structured agent organizations. The TÆMS and GPGP paradigms that had been developed and used to represent an organizational design (Decker and Lesser 1994), were used to develop a distributed data processing organization. The authors demonstrated that the grammar based language enabled the designer to model task structures in a suitable manner that otherwise had generated unexpected behaviour from the GPGP coordination mechanisms.

2.4. Project Management

A domain independent decision maintenance server was described and proposed as a mechanism for federating heterogeneous design agents (Petrie 1993). One of the issues dealt with within this paper was *semantic unification*, or the ability of one system to understand the language and knowledge of another. Petrie also discussed the propagation of change within a general design model. The Knowledge Query and Manipulation Language (KQML) developed by DARPA provided a degree of semantic unification by producing a set of performatives or conversation rules that the various agents must agree upon (Finin et al. 1992). Petrie used a subset of REDUX to add semantics to the KQML primitives to provide functionality with a decision maintenance server. It was subsequently recognised that the Redux model could be regarded as a reusable ontology of a similar kind as that represented by Ontolingua (Gruber 1992), and with a similar notion as that developed with the Agent Building Shell (Barbuceanu 1996a). It differed, however, in that it was not an ontology of design domain objects, but an ontology that could be used to encapsulate such objects. The Redux' ontology essentially described a design decision as consisting of a goal (or objective or task) that it reduced (satisfied or accomplished) and a result.

Petrie et al. recognised that traditional project management techniques were not sufficient for managing the many tasks in the design and development process (Petrie et al. 1998). The most significant

concern of Petrie's was that these systems did not provide the proper mechanisms for notifying the correct people of the effects of change at the correct time in the process. Petrie noticed that even in small design projects, people lose their ability to maintain a comprehensive picture of the history and interplay of design decisions, constraints and rationales.

In an attempt to maintain a comprehensive representation of the design problem, many more people are often notified of a design change than is actually necessary, thus burdening the whole design process. This philosophy of excessive communication may be seen within the PACT experiments. Here the focus was aimed more towards cooperation than coordination, with the notification of a design change being broadcast to all of the agents within the community. The agents that were then responsible for undertaking design activity based upon the decision could subsequently carry out their activity, however, for larger teams of multi-disciplinary design agents, the method would result with excessive unnecessary communication. This problem was tackled within the SHADE project (McGuire et al. 1993) with content based routing, which was a mechanism to ensure that only the people interested in the information would receive it.

A subset of the REDUX model, called Redux' was used to manage the propagation of change to facilitate the development of an integrated project management system called Procura (Goldmann 1996; Petrie et al. 1998). The methodology differed from general workflow management systems due to the interleaving of the design and construction planning, allowing both the design and plan to be changed as necessary. The requirement for this ability resulted from the possibility of a design change affecting the design, as well as the design plan and schedule. The Procura model of process coordination shared some similarities with workflow management, but it did not require the identification of all of the tasks and ways of doing them before process execution. The techniques used to develop Procura were further developed within CoMoKit (Dellen and Maurer 1996).

Petrie et al. (1995) also demonstrated the use of Redux' to coordinate the design process using the concept of Pareto optimality within a framework called Next-Link. Petrie recognised the need to provide a method that would enable the quality of the design artefact to be measured. This need would be increased within a design scenario that included multiple designers and engineers working within multiple domains of a single design artefact. Such a method would enable the designer to determine whether a particular design change has improved or reduced the quality of the design. The concept of Pareto optimality was used since it was a general technique that enabled multiple, often conflicting objectives and constraints to be considered. The Redux' design model provided a formal notion of an objective that would be required to implement such a principle. Pareto optimality moved away from having a single 'weighted' objective that was a function of all of the objectives, to providing a measurement of how close the solution was to the ideal scenario. The tracking of Pareto optimality enabled some aspects of dependency-directed backtracking to be incorporated. 'Tracking' meant that the problem solver would be automatically notified of Pareto optimality loss and of the particular opportunity to improve the design. Providing such functionality enabled opportunities to improve local solutions that otherwise might have been lost, and reduced 'thrashing' or performing design activity that would return to previously explored designs that were known to be sub-optimal.

Three strategies were adopted to facilitate Concurrent Simultaneous Engineering (CSE) of the product development process within the CONSENS project; parallelization, standardization and integration (Bullinger and Warschat 1996). Parallelization was achieved by allowing succeeding processes to be started prior to the completion of preceding processes. The authors recognized however that providing concurrency in this manner would increase the amount of uncertain and incomplete information. Improved coordination was achieved through standardisation by avoiding repetition and needless work. Integration within CONSENS allowed interdisciplinary teams to cooperate to solve a common objective. To facilitate parallelization, standardization and integration within a CSE context the authors proposed a framework to model, support, control and integrate processes and teams, an information management system to manage, change and release product data, and a product information archive to store and access a common product model. A number of systems were developed to realize these aspects of CSE including a STEP based PDM system, a design management system and a process modelling system. The authors, however, realised that an implementation of CSE would require more than just a set of supporting tools, but ''a new orientation of the complete product-development process''. The CONSENS framework was implemented within a number of European companies.

Van Den Hamer and Lepoeter (1996) identified 'five orthogonal dimensions' that are fundamental to Design Data Management. The authors suggest that each dimension, when considered separately is

relatively simple to implement, however two or more need to be considered to solve real-world applications. A discussion follows of the version, views, hierarchy, status and variants dimensions, and the interactions between them and the systems developed within two dimensions. The authors argue that the systems that have been developed to consider more than three dimensions have had the other dimensions tagged on as an afterthought, and that systems may have limited applicability across disciplines due to the variation in the importance of the dimensions.

Bilgiç and Rock (1997) discussed the state-of-the-art of Product Data Management (PDM) systems and described the development of a knowledge aided design system that was used to manage data and processes related to the product development lifecycle. The PDM system view of the design process was described as ''providing the mechanisms that enable the right data to be made available to the right person at the right time''. Bilgiç identified a number of the weaknesses of PDM systems resulting from the management of documents and data, and described how the management of knowledge would provide a more suitable and flexible solution for systems management.

A discussion of the practicalities of using classical data models (e.g. relational, network, and hierarchical) for the modelling of complex product and process data lead to the conclusion that these types of model were too simple and due to their levels of abstraction were limited in their ability to capture the diversity of information within engineering applications (Chen and Hsiao 1997). The authors describe the development of an object-oriented framework for the management and coordination of concurrent engineering processes. After reviewing the product delivery process, the functional requirements for team data management were identified as:

- a structure to define the components within a product and their relationships,
- methods that enable a project to be configured with respect to members, activities, roles etc.,
- the ability for team members to manage product and process items,
- the management of release, change and notification.

The authors finally discuss the application of the framework within the collaborative data management of a transmission system.

Peng and Trappey (1998) also mention the limitations of a number of Engineering Data Management (EDM) modelling systems with respect to support for decision making, data and process representation and life-phase modelling. The authors describe the use of the STEP product data representation and exchange standard within the development of a number of data models of an EDM system whilst attempting to overcome the limitations of the other modeling systems. The STEP data exchange standard uses the EXPRESS object-oriented language to enable product data to be represented within a neutral format facilitating the integration of existing legacy software. Six main functions were identified for an EDM system to successfully manage product-related information throughout the lifecycle; engineering drawing management, material specification management, product structure management, production schedule management, technical document management, and engineering change management. The system enabled distributed CAD/CAM applications to have controlled access to the relevant information within the EDM system.

2.5. Workflow Management

Workflow management systems have generally been developed for application within business processes by modelling the activities, flow of data, organizational structure and tools available within a business.

Mohan et al. (1995) described the Exotica workflow management research project, part of which was based on the development of mechanisms that would allow a distributed workflow to continue in the event of a failure. Workflows have typically been defined in advance which has meant that many error conditions have had to be defined to enable the system to continue in the event of failure. Mohan dealt with errors using 'forward recovery' enabling the process to progress. Distributed and mobile computing were also discussed enabling disconnected clients to execute parts of the process without being connected to a centralized server.

Jennings et al. (1996) described the ADEPT agent-based framework for managing business processes. The agents within ADEPT were based upon the GRATE and ARCHON agent models, hence the enactment of the workflow model was again distributed rather than centralized. The use of an agent architecture for workflow management was qualified by the inherent distribution of data and problem solving capabilities within organizations, the need for sophisticated interaction and negotiation, and the inability to prescribe the problem from start to finish. The contract net protocol was used as the foundation for its negotiation mechanism which was extended to enable business process management interactions to be expressed more clearly.

The ADEPT workflow model was extended within ADEPTflex which provided the functionality to change the structure of a workflow model during execution (Reichert et al. 1998).

The use of workflow scripts within distributed systems has been the focus of a number of research groups (Ranno et al. 1997; Dellen et al. 1997) with the realization that the scripts, like the systems, need to be dynamic in nature. Ranno et al. developed and implemented a coordination language for expressing the composition and dependencies of distributed applications. Workflow scripts were defined using the coordination language which specified the tasks to be undertaken according to their dependencies. The dynamic nature of distributed systems was dealt with by providing mechanisms that allowed the workflow systems to adapt and change their internal structures by the addition or removal of tasks and dependencies.

Dellen et al. produced a workflow management methodology that could be applied within a dynamic environment such as an engineering project. One of the shortcomings of workflow methods that was tackled was to allow the model to change as the design process progresses. The need to manage change was also identified as a requirement for a system to support engineering projects which was supported with an application of the Redux' model (Petrie 1993).

2.6. Task Models

The TÆMS framework enables complex, computationally intensive task environments to be modelled and simulated such that computational theories of coordination may be built and tested (Decker 1996). TÆMS differs from other coordination representations that are concerned with a single decision, by representing the possible outcomes of many sequences of interrelated choices. Decker recognised that TÆMS provided much of the information that would enable the framework to be used as a scheduling device, and would provide that functionality when working with the GPGP framework, however this was not its intended purpose.

Coordination theories may be developed using TÆMS by providing information regarding the structure and other characteristics of tasks in an environment and the information that is known and actions that can be undertaken by agents in certain organizational roles.

A multi-agent framework called MADEsmart demonstrated the use of Decker's Generalised Partial Global Planning algorithms (Decker and Lesser 1995) to coordinate the design activity of a number of design groups during the preliminary design process (Jha et al. 1998). A number of different agent types were developed to manage particular aspects of the design process. The results of the DARPA work was again used as the basis of the development of the communication of knowledge between the agents. MADEsmart also utilized the TÆMS representation of task structures to enable the formal coordination of design activity.

Barrett et al. (1997) also described the development of the MADEsmart environment which provided task coordination, multi-disciplinary optimization and easy access to information. The goal of this research was to enable the rapid exploration of design alternatives within the aircraft industry. The MADEsmart environment used an intelligent agent architecture to automate processes and coordinate tasks. Natural language processing was used to capture text-based design requirements and specifications and ensure that this information would be distributed to the users that required it. Such a mechanism could also be used to inform users of any conflicts as they arise between requirements and specifications, and proposed design solutions. The coordination structure used within the agents was again centred on the TÆMS and GPGP paradigms. Evaluation mechanisms were produced based upon the reduction in time required to produce engineering drawings and the number of drawings.

Bilgiç described the development of the Systems Design Management Agent (SDMA) within the MADEsmart framework which was concerned with reasoning about agent interaction, the design process, or the environment (Bilgiç 1997). Risks were broadly defined by Bilgiç as *events that cause a delay in the project, events that cause cost overruns, and, events that cause deficiencies in technological performance*. Bilgiç subsequently described the requirements for a technology to be able to acquire, represent and reason on knowledge about different types of risk.

3 Concluding Remarks

It is apparent that one of the major driving forces for coordination is the increased use of distributed design activity. This has been duplicated within the computer domain with the development of distributed agents operating within distinct disciplines performing tasks which may be viewed as being part of an encompassing design process. One of the benefits of providing a distributed agent architecture within a computerised environment is that the entire design problem need not be encompassed within a single

piece of software, rather, existing legacy code may be used to perform pre-defined tasks under the supervision of a software agent. Managing a number of these software agents however requires a formalised approach such that the activity is performed in a coordinated and efficient manner.

Coordination has been reviewed from a number of different perspectives. Various mechanisms have been described with the general purpose of managing and representing generic coordination knowledge whilst being sufficiently expressive to manage domain specific knowledge. Experimental testbeds have been developed to enable fundamental coordination mechanisms to be tested for validity by observing the activity of a given simulated environment. No indication is given, however, of how applicable these mechanisms may be within a domain such as engineering design and it is believed that such an application may require domain specific coordination information for the process to operate correctly.

Various languages have been developed which contain coordination knowledge as their content, to enable agents to schedule their activities based upon the requests of other agents. The agents tend to start with only local knowledge and gradually build up their representation such that they may perform acceptably within a community. This seems to rely on the assumption that it is not clear what is happening from the start, and it is up to the agents to try to work it out, an assumption that would seem to be unacceptable within an engineering environment.

What is apparent, from the research reviewed is that there is no clear way of managing coordination, and despite the many attempts to generalise it, such that it may be applied to any domain, it is debatable whether the rules obtained and used within one domain may be of any use within another.

Acknowledgements

The authors gratefully acknowledge the support given by the Engineering and Physical Science Research Council who provided the grant RES/4741/0929 that enabled this work to be undertaken.

References

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, Glasgow, pp 151–174

Barbuceanu M (1995) COOL: A Language for Representing and Executing Coordinated Behaviour in Multi- Agent Systems. ICMAS Proc., San Francisco, CA, pp 17–24

Barbuceanu M, Fox MS (1996a) The Agent Building Shell: A Tool for Building Enterprise Multi-Agent Systems. Canadian Artificial Intelligence, 40:9–11

Barbuceanu M, Fox MS (1996b) The Design of a Coordination Language for Multi-Agent Systems. Lecture Notes in Computer Science, European Conference on Artificial Intelligence, Budapest, pp 341–356

Barrett T, Coen G, Hirsh J et al. (1997) MADEsmart An Integrated Design Environment. Proceedings of DETC97, ASME Design Engineering Conferences, Sacramento, CA

Bilgiç T, Rock D (1997) Product Data Management Systems: State-Of-The-Art and the Future. Proceedings of DETC'97, ASME Design Engineering Conferences, Sacramento, CA

Bilgiç T (1997) Systems Management in Concurrent Engineering using Intelligent Software Agents. Proceedings of ISCIS'97, 12th International Symposium on Computer and Information Sciences, Antalya, Turkey

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

Chen Y-M, Hsiao, Y-T (1997) A collaborative data management framework for concurrent product and process development. International Journal of Computer Integrated Manufacturing, 10(6):446–469

Coates G, Whitfield RI, Duffy AHB et al. (2000) A review of coordination approaches and systems – Part II: An operational perspective. To be published in Res Eng Design Vol 12

Cutkosky MR, Engelmore RS, Fikes RE et al. (1993) PACT: An experiment in integrating concurrent engineering systems. Computer, 20:28–37

Cutkosky MR, Tenenbaum JM, Glicksman J (1996) Madefast: An exercise in collaborative engineering over the Internet. Comm ACM, 39(9):78–87

Decker K (1996) Task environment centred simulation. In: Prietula M, Carley K, Gasser L (eds), Simulating Organizations: Computational Models of Institutions and Groups. AAAI Press/MIT Press

Decker K, Lesser V (1992) Generalising the partial global planning algorithm. Int J Intelligent Cooperative Infor Syst 1(2):319–346

Decker K, Lesser V (1994) Task environment centred design of organizations. AAAI Spring Symposium on Computational Organization Design, Stanford, CA

Decker K, Lesser V (1995) Designing a family of coordination algorithms. Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95, San Francisco, CA

Dellen B, Maurer F (1996) Dynamic modelling of design processes. 2nd Knowledge Engineering Forum, in Bericht 01/96 des SFB

Dellen B, Maurer F, Pews G (1997) Knowledge based techniques to increase the flexibility of workflow management. Data & Knowledge Eng J

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

Durfee EH, Lesser VR, Corkill DD (1989) Trends in cooperative distributed problem solving. IEEE Trans on Knowledge and Data Eng 1(1):63–83

Durfee EH, Montgomery TA (1989) MICE: A Flexible Testbed for Intelligent Coordination Experiments. Proceedings of the Ninth Workshop on Distributed Artificial Intelligence, Eastsound, WA, pp 25–40

Durfee EH, Lesser VR (1991) Partial global planning: A coordination framework for distributed hypothesis forma-

tion. IEEE Trans Systems, Man and Cybernetics 21(5):1167–1183
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)
Finger S, Fox MS, Prinz FB et al. (1992) Concurrent design. Applied Artificial Intelligence 6:257–283
Finin T, Weber J, Wiederhold G et al. (1992) DRAFT Specification of the KQML Agent Communication Language. Official Document of the DARPA Knowledge Sharing Initiative's External Interfaces Working Group, Enterprise Integration Technologies, Inc. Tech. Report 92–04
Fox MS, Barbuceanu M, Gruninger M (1996) An organization ontology for enterprise modelling: preliminary concepts for linking structure and behaviour. Computers in Industry 29:123–134
Goldmann S (1996) Procura: A Project Management Model of Concurrent Planning and Design. Proceedings of WETICE-96, Stanford, CA
Gruber TR (1992) Ontolingua: A Mechanism to Support Portable Ontologies. Stanford University, Knowledge Systems Laboratory, Technical Report KSL 91–66
Gruber TR, Tenenbaum JM, Weber JC (1992) Toward a knowledge medium for collaborative product development. In: Gero JS (ed), Artificial Intelligence in Design'92, Kluwer Academic, Boston, pp 413–432
Gupta L, Chionglo J, Fox MS (1996) A constraint based model of coordination in concurrent design projects. Proceedings of WETICE-96, Stanford, CA
Hogg T, Huberman BA (1991) Controlling chaos in distributed systems. IEEE Trans Systems, Man and Cybernetics 21(6):1325–1332
de Hoog R, Benus B, Metselaar C et al. (1994a) Organisation model: Model definition document. Deliverable DM6.2c, ESPRIT Project P5248 KADS-II/M6/M/UvA/041/3.0, University of Amsterdam and Cap Programmator
de Hoog R, Martil R, Wielinga B et al. (1994b) The Common KADS model set. Deliverable DM1.1c of ESPRIT Project P5248 KADS-II, University of Amsterdam and Cap Programmator
Jennings NR, Mamdani EH, Laresgoiti I et al. (1992) GRATE: A general framework for cooperative problem solving. IEE-BCS J Intelligent Systems Eng 1(2):102–114
Jennings NR (1992) Towards a cooperation knowledge level for collaborative problem solving. Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92), Vienna, Austria, pp 224–228
Jennings NR, Pople JA (1993) Design and implementation of ARCHON's Coordination Module. Proceedings of the Workshop on Cooperating Knowledge Based Systems, Keele, UK, pp 61–82
Jennings NR (1996) Coordination techniques for distributed artificial intelligence. In: O'Hare GMP, Jennings NR (eds), Foundations of Distributed Artificial Intelligence, Wiley, pp 187–210
Jennings NR, Faratin P, Johnson MJ et al. (1996) Agent-based business process management. Int J Cooperative Infor Syst 5(2&3):105–130
Jha KN, Morris A, Mytych E et al. (1998) Agent support for collaborative design. Proceedings of DETC98, 1998 ASME Design Engineering Conferences, Atlanta, GA
Klein M (1995) iDCSS: Integrating workflow, conflict and rationale-based concurrent Engineering coordination technologies. J Concurrent Eng Res and Applic 3(1):21–29
Lesser VR, Corkill DD (1983) The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks. AI Magazine 4(3):15–33
Lesser V, Decker N, Carver A et al. (1998) Evolution of the GPGP Domain-Independent Coordination Framework. University of Massachusetts Computer Science Technical Report 1998–05
Lesser V (1998) Reflections on the Nature of Multi-Agent Coordination and its Implications for an Agent Architecture. Autonomous Agents and Multi-Agent Systems, Kluwer Academic, 1:89–111
McGuire JG, Kuokka DR, Weber JC et al. (1993) SHADE: Technology for knowledge-based collaborative engineering. J Concurrent Eng Res and Applic 1(2)
Mohan C, Alonso G, Guenthoer R et al. (1995) An overview of the Exotica Research Project on workflow management systems. Proceedings of the 6th International High Performance Transactions Systems Workshop (HPTS), Asilomar, CA
Nagendra Prasad MV, Decker K, Garvey A et al. (1996) Exploring organizational designs with TAEMS: A case study of distributed data processing. Proceedings of the Second International Conference on Multi-Agent Systems, AAAI Press, Kyoto, Japan
Patil RS, Fikes RE, Patel-Schneider PF et al. (1992) The DARPA Knowledge Sharing Effort: Progress Report. In: Nebel B, Rich C and Swartout W (eds), Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, Cambridge, MA, pp 777–788
Peng T-K, Trappey AJC (1998) A step toward STEP-compatible engineering data management: the data models of product structure and engineering changes. Robotics and Computer-Integrated Manufacturing 14:89–109
Petrie C (1993) The Redux' Server. Proceedings of the First International Conference on Intelligent and Cooperative Information Systems (ICICIS),Rotterdam, Netherlands, pp 134–143
Petrie C, Webster TA, Cutkosky MR (1995) Using Pareto Optimality to coordinate distributed agents. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 9:261–281
Petrie C, Jeon H, Cutkosky M (1997) Combining constraint propagation and backtracking for distributed engineering. ECAI-96 Workshop on Non-Standard Constraint Processing, Budapest, August (revised for AAAI-97 Workshop on Constraints and Agents, Providence, RI, July 1997)
Petrie C, Goldmann S, Raquet A (1998) Agent based project management. Proceedings of the International Workshop on Intelligent Agents in CSCW, Dortmound, pp 1–17
Ranno F, Shrivastava SK, Wheater SM (1997) A system for specifying and coordinating the execution of reliable distributed applications. Conference on Distributed Applications and Interoperable Systems (DAIS'97), Cottbus, Germany
Reichert M, Dadam P (1998) ADEPTflex – Supporting Dynamic Changes of Workflows Without Losing Control. J Intelligent Infor Syst (JIIS), Special Issue on Workflow Management Systems, 10(2):93–129
Smith RG (1980) The Contract Net Protocol: High-level communication and control in a distributed problem solver. IEEE Trans Computers C-29(12):1104–1113
So YP, Durfee EH (1994) Modelling and designing computational organizations – An organizational self- design model

for organizational change. Proceedings of the 1994 AAAI Computational Organisation Design Symposium, Stanford, MA

So YP, Durfee EH (1996) Designing tree-structured organizations for computational agents. Computational and Mathematical Organization Theory 2(3):219–246

Toye G, Tenenbaum JM, Cutkosky MR et al. (1994) SHARE: A methodology and environment for collaborative product development. Int J Intelligent and Cooperative Infor Syst

Van Den Hamer P, Lepoeter K (1996) Managing design data: The five dimensions of CAD frameworks, configuration management, and product data management. Proc IEEE 84(1):42–56

Weih HP, Schü J, Calmet J (1994) CommonKADS and cooperating knowledge based systems. Proceedings of the 4th KADS User Meeting, Bonn

Wittig T, Jennings NR, Mamdani EH (1994) ARCHON – A Framework for Intelligent Co-operation. IEE-BCS J Intelligent Syst Eng, Special Issue on Real-time Intelligent Systems in ESPRIT, 3(3):168–179

Appendix A

Table 1. Coordination systems – summary of features.

System	Coordination Type	Knowledge Type	Techniques Used	Application
ADEPT	Federated architecture.	General.	As GRATE & ARCHON, negotiation.	Management of business processes.
ARCHON	Federated architecture.	Generic knowledge of cooperation and situation assessment.	Social interaction management, tracking of cooperative activities, cooperative reasoning.	Electricity distribution and transportation management, cement factory control.
CommonKADS	Organisational.	Organisation, task, agent, communication, expertise.	Knowledge acquisition, Contract Net protocol.	Not stated.
COOL	Language	Domain, environment and self.	Knowledge management, cooperative conflict management, ontologies.	Supply chain integration.
GPGP	Algorithm	– Coordination algorithms used within PGP generalised and applied to the same problem. –		
GRATE	Federated architecture.	State, capability, intention & evaluation.	Integrative (uses pre- existing intelligent systems), knowledge bases.	Industrial process control.
iDCSS	Environment	Rationale, workflow, conflicts.	Process management, conflict management, rationale capture.	Concurrent engineering.
MADEsmart	Federated architecture.	Task-structure	TÆMS & GPGP, scheduling.	Design of aircraft wing parts.
MICE	Testbed	Environmental constraints and characteristics.	Experimental testbed, artificial intelligence.	Forest-fire fighting, cooperative robotic reconnaissance.
PACT	Federated architecture.	Shared design knowledge.	Development of communication protocols, knowledge sharing, negotiation.	Design and simulation of a robotic manipulator.
PGP	Algorithm	Local and partially global activities.	Group problem solving, planning, contracting, result sharing.	Simulated vehicle monitoring.
Redux'	Enables federated architectures.	General.	Decision maintenance, management of the propagation of change, optimality and backtracking.	Re-implementation of PACT example.
SHADE	Federated architecture.	Shared design knowledge.	Shared knowledge communication medium, knowledge encapsulation, content-based routing.	General engineering design applicability.
SHARE	Environment.	Shared design knowledge.	Internet-based knowledge capture and management.	General engineering design applicability.
TÆMS	Algorithm	Task environments, structures and interrelationships, environmental constraints	Modelling of task quality and time taken.	Distributed situation assessment, hospital patient scheduling, airport ground services management.
TOVE	Organisational.	Agent, role, position, goal, communication, authority, commitment.	Ontology design.	Not stated.