View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Strathclyde Institutional Repository

Strathprints Institutional Repository

Coates, G. and Duffy, A.H.B. and Whitfield, I. and Hills, W. (2003) *An integrated agent-oriented approach to real-time operational design coordination.* AI EDAM - Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 17 (4). pp. 287-311. ISSN 0890-0604

Strathprints is designed to allow users to access the research output of the University of Strathclyde. Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. You may not engage in further distribution of the material for any profitmaking activities or any commercial gain. You may freely distribute both the url (http://strathprints.strath.ac.uk/) and the content of this paper for research or study, educational, or not-for-profit purposes without prior permission or charge.

Any correspondence concerning this service should be sent to Strathprints administrator: mailto:strathprints@strath.ac.uk

http://strathprints.strath.ac.uk/

Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2003), *17*, 287–311. Printed in the USA.
Copyright © 2003 Cambridge University Press 0890-0604/03 $16.00
DOI: 10.1017/S0890060403174021

An integrated agent-oriented approach to real-time operational design coordination

GRAHAM COATES,[1] ALEX H.B. DUFFY,[2] IAN WHITFIELD,[2] AND WILLIAM HILLS[1]

[1]Engineering Design Centre, University of Newcastle, Newcastle upon Tyne, United Kingdom
[2]CAD Centre, University of Strathclyde, Strathclyde, United Kingdom

(RECEIVED July 17, 2002; ACCEPTED September 15, 2003)

Abstract

Within the engineering design community there is support for further research into the development of improved approaches to design management. Such research has lead to coordination being identified as an important and pervasive characteristic of many existing approaches (e.g., concurrent engineering and work-flow management). In this article, operational design coordination is proposed as the basis for an improved approach. This article also presents a novel integrated approach that incorporates the key elements of operational design coordination: coherence, communication, task management, resource management, schedule management, and real-time support. Through unifying these key elements, this approach provides an integrated means of managing design in a controlled and harmonious fashion. The approach also provides knowledge of the constituent techniques involved in operational design coordination, the interrelationships and dynamic interactions between them, and the knowledge used and maintained within and between them. The approach has been realized within an agent-oriented system called the Design Coordination System, which provides a systematic means of simultaneously coordinating operational management tasks and technical design tasks. To evaluate the approach, the system has been applied to an industrial case study involving the computational process of turbine blade design. This application has been shown to enable the structured undertaking of interrelated tasks by allocating and using resources of varying performance efficiency in an optimized fashion in accordance with dynamically derived schedules in a coherent, appropriate, and timely manner. This is achieved by managing tasks, their dependencies, and the information required to undertake them. In addition, the approach enables and sustains the continuous optimized use of resources by monitoring, forecasting, and disseminating resource performance efficiency. The approach facilitates dynamic scheduling and the subsequent enactment of the resulting schedules. Decision making for rescheduling is also incorporated within the approach such that it is only performed as and when appropriate. If rescheduling is performed, it is done so in parallel with task enactment such that resources continue to be utilized in an optimized manner.

Keywords: Agent-Oriented Approach; Design Management; Real-Time Coordination

1. INTRODUCTION

Competitive pressure compels engineering companies to out perform their contemporaries in order to be more attractive to existing and potential customers. Wallace (1987) indicated that in order to maintain continuing competitive advantage, senior management in manufacturing industries should coordinate and control personnel to fulfill the main business activities, which include design and development. In the context of engineering design, Andreasen et al. (1996) recognized that it is increasingly evident that significant improvements and efficiency gains can be made because much time and effort is lost as a result of the lack of focus on the management of design work.

In 1916, Fayol (1949) wrote *General and Industrial Management*, in which management was described as a process consisting of planning, organization, coordinating, directing, and controlling. Lock (1993) named Fayol the founding father of engineering management and modern management theory. In addition, Bennett (1996) cited Fay-

Reprint requests to: Graham Coates, School of Engineering, University of Durham, Durham, DH1 3LE, United Kingdom. E-mail: graham.coates@durham.ac.uk

ol's work as forming the origin of the field of the management process, shaping the basis for much other work in this area. Despite Fayol's pioneering work on management in the early 1900s, only in recent years has engineering management started to attain the status of a recognized discipline (Lock, 1993). Despite this recognition, research efforts in engineering management have been described as fragmented and uncoordinated. Furthermore, Lock noted that in the current climate of rapid technological change and an intensively competitive global environment, there is a demand for a renewed emphasis on effective engineering management and a reevaluation of traditional attitudes and approaches. This point is echoed by Thamhain (1992), who also recognized that today's engineering environment is more challenging than ever before because of increased technical complexity and the interdependency of technical tasks.

Management has been considered to comprise a strategic level and an operational level (Greenley, 1989; Cole, 1994), and Finlay (2000) also noted that an organization consists of a number of parts that includes a strategic apex to oversee the whole of the business and an operational core, described as the people who perform the basic, day to day processes. Greenley (1989) indicated that strategic management provided a framework for operational management, which was described as being concerned with the efficient use of the existing production capacity. Similarly, Cole (1994) stated that "strategic management produces the primary goals and framework within which they can be realized for operational management." Furthermore, it was noted that the concerns of strategy were effectiveness (i.e., ensuring that the organization is doing the right things), whereas the concerns of operations were efficiency (i.e., doing things right). As such, the performance of the design development process can be improved through both the strategic and operational levels of management. The work presented in this article is aimed at design coordination at the operational level of management only. However, research on design coordination at the strategic level of management has been conducted in collaboration with this work (Whitfield et al., 2000*a*, 2000*b*).

From an operational perspective, management of the design development process of large, made to order products can be complex, expensive, and time consuming because of the involvement of many resources and tasks and of large quantities of data, information, and knowledge. This complexity is further compounded by the fact that resources are often skilled in a variety of disciplines and exhibit varying proficiency regarding the completion of multiple interrelated tasks. Furthermore, because of unforeseen circumstances, resources may not perform as intended or scheduled tasks may not progress as expected, the outcome of which will influence the performance of the design development process.

A well-organized approach to the design development process lies at the heart of an effective engineering company because it can enable the reduction of cost and time while meeting customer quality requirements. Thus, to remain competitive, new approaches to managing the design development process are needed to ensure efficient processes. Indeed, the latter part of the 20th century saw the introduction of an increasing number of new management initiatives or philosophies aimed at improving the competitiveness of companies. Engineering design has seen the advent of a range of management approaches that have been implemented within industry. Coordination has been observed as an important and pervasive characteristic within a number of these management approaches, such as models of the engineering design process (Ray, 1985; Cross, 1994), concurrent engineering (Duffy et al., 1993; McCord & Eppinger, 1993; Prasad, 1996; Tan et al., 1996; Perrin, 1997; Coates et al., 1999), work-flow management (Alonso et al., 1996; Yu, 1996; Piccinelli, 1998; Du & Shan, 1999), project management (Oberlender, 1993; Bailetti et al., 1994; Cleetus et al., 1996; Lock, 1996; Bendeck et al., 1998), design integration (Hansen, 1995), and computer supported cooperative work (Malone & Crowston, 1994; Schal, 1996).

Despite being widely cited as an important characteristic of the approaches mentioned, the understanding of coordination conveyed varies considerably. The existence of varying perceptions of coordination has lead to the recognition that there is a requirement for further research in this field with the aim of gaining a better understanding of its nature and potential as an approach to engineering management in its own right. Indeed, Duffy et al. (1999) indicated that there is a growing interest within academia in calling for further research in the area of design coordination. In response to this call, an extensive review of literature has been conducted that draws on perceptions from several disciplines; namely, engineering design, distributed artificial intelligence, and organizational theory (Coates et al., 2000; Coates, 2001). On the basis of these reviews, the key elements of operational design coordination have been established as coherence (Durfee & Montgomery, 1990; Jennings, 1996; Wilson & Shi, 1996; de Jong, 1997; Jamali et al., 1999), communication (Kleinman, 1990; Findler & Elder, 1995; Cleetus et al., 1996; de Jong, 1997; Hayden et al., 1999), task management (Kusiak & Wang, 1993; Duffy et al., 1994; Eppinger et al., 1994; Malone & Crowston, 1994; Decker & Lesser, 1995), schedule management (Ray, 1985; Malone, 1987; Dellen & Maurer, 1996; Bendeck et al., 1998; Lesser, 1998), and resource management (MacCallum & Carter, 1991; Duffy et al., 1993; Andreasen et al., 1996; Davis & Sydir, 1996; Durfee & So, 1997).

With consideration of the various interpretations of the authors named above, the key elements of operational design coordination can be defined as follows:

- *coherence:* integrating, or linking together, resource efforts and tasks in a harmonious manner to avoid chaos,
- *communication:* interactions involving the exchange of structured and meaningful data, information, and knowledge,

- *task management:* that is, the organization and control of tasks, and the dependencies between them, such that they can be undertaken and completed in a structured manner,
- *schedule management:* that is, managing the dynamic assignment of tasks to resources, and the enactment of the resulting schedules, and
- *resource management:* that is, organizing and controlling resources to enable their continuous optimized utilization.

It has been recognized that engineering design is changeable as a result of the evolution of the multidisciplinary groups, activities, and information involved (Andreasen et al., 1996; Duffy, 1998). Thus, a further key element of operational design coordination is identified and can be defined as

- *real-time support:* how to manage and adapt to a changeable (i.e., dynamic and unpredictable) process.

Furthermore, we know that there is a requirement for an approach to operational design coordination that integrates the six key elements identified. Such an integrated approach will provide an original and significant contribution to knowledge in the field of operational engineering management by allowing design to be coordinated in a coherent manner. This article presents such an approach that provides knowledge of the constituent techniques of operational design coordination, the interrelationships and dynamic interactions between the techniques, and the information used and maintained within and between the techniques. As such, it is not only do the individual techniques themselves that define the approach but also, more significantly, the interrelationships and interactions that enable them to be integrated. Indeed, Harrison (1992) recognized that it is the notion of an encompassing approach that is more important than the specific groups of techniques used.

2. RELATED WORK

A number of concurrent engineering-based approaches to design management focus on managing tasks (i.e., sequencing tasks according to their dependencies). These approaches involve, for example, decomposing a product or system into sets of tasks and then representing their interactions using a nonstructured matrix (Kusiak & Park, 1990; Pourbabai & Pecht, 1994). Subsequently, techniques are used to transform the matrix to enable the detection of groups of tasks that may be scheduled and performed simultaneously. Similarly, Eppinger (2001) employs the design structure matrix (DSM) to make a product development process more efficient by reducing iteration, which wastes time and resources. The DSM represents the tasks required to develop a product and the information flow between them. A variety of techniques are then used to optimize information flow (i.e., to reduce iteration). Techniques such as partitioning, resequencing, decoupling, and clustering can be used for the purposes mentioned above (Kusiak & Wang, 1991; Eppinger et al., 1994; Pimmler & Eppinger, 1994).

Several coordination-based systems are aimed at the incremental revision/development of project plans and schedules. Within these systems, management activities such as planning, scheduling, and enactment are *interleaved* because of the occurrence of changes or decisions being made during the project as more information becomes available. For instance, CoMo-Kit supports project planning and coordination for complex, distributed design projects/development processes by alternating, planning, and enactment (Dellen & Maurer, 1996). In addition, Procura is a project management model that allows planning and scheduling agent-based design projects to occur concurrently (Goldmann, 1996). Similarly, the system architecture presented by Bendeck et al. (1998) supports the coordination of management activities in the software development process by interleaving planning, scheduling, and enactment. MIDAS (Manufacturing Integration and Design Automation System) is a distributed environment infrastructure for the planning and execution of design and manufacturing processes (Kwon et al., 2002). A control mechanism and a common communication medium enable users of the system to share information in a distributed environment such that design and manufacturing activities can be carried out collaboratively.

A number of approaches are oriented toward coordinating and managing task agendas of human or computational agents. For instance, Decker and Lesser's (1995) support tool for distributed, cooperative work consists of computational agents assisting people in coordinating their activities by managing their agenda. Situated at each user's workstation, agents offer task orderings according to user preferences and provide agenda management to coordinate computational agents according to these preferences. As such, a distributed coordination process occurs, and agendas are produced in a collaborative manner. Similarly, PACT (Project Assessment and Coordination for Teams) is aimed at managing projects and coordinating people (Cleetus et al., 1996). PACT is a multiuser system with a communication interface enabling project team members to notify or query others regarding tasks. The ability to constantly be aware of each other's activities provides the mechanism to coordinate people.

Systems have been developed to assist in the arrangement of meetings (Jennings & Jackson, 1995) and the control of meetings (Pena–Mora et al., 2000). Jennings and Jackson (1995) present an agent-based distributed meeting scheduling system. Knowledge of the preferences and commitments of each user of the system are used by their respective Meeting Scheduling Agent to arrange meetings. Pena–Mora et al. (2000) present a conferencing architecture for managing designers and engineers in a distributed design meeting, called CAIRO (Collaborative Agent Interaction and Synchronization). The top-level system architecture of CAIRO includes a directory of meeting controllers and designers and engineers using the system and includes one

Collaboration Manager per participant. The CAIRO system employs a number of control strategies that are used to decide which participant may contribute to a conference at a given time. Software agents are said to be a further component of CAIRO, as they work alongside designers and engineers using the system to make the meeting effective and productive through agenda and time management and through proposing suitable control strategies given the nature of the meeting.

The Virtual Design Team (VDT) is presented as a computational simulation model for project organizations (Jin & Levitt, 1996). In the context of the VDT, an organization is viewed as "an information-processing and communication system, structured to achieve a specific set of tasks, and composed of limited teams (called actors) that process information." As such, the VDT model consists of tasks, actors, communication tools, and an organization structure. The VDT simulation identifies significant information-flow bottlenecks involving activities and actors (Kunz et al., 1998), which are resolved by users of the VDT.

The approaches mentioned provide valuable contributions in the field of operational engineering design management. However, although these approaches recognize some of the key elements of operational design coordination (i.e., coherence, communication, task management, resource management, schedule management, and real-time support), no single approach integrates all of them by incorporating the appropriate techniques and knowledge of the interrelationships between them. Indeed, a detailed critical review of existing approaches related to operational engineering management with respect to the key elements of operational design coordination has shown the need for an integrated approach (Coates, 2001). As such, the aim of this article is to present a novel approach that integrates all six key elements identified. The approach developed has been realized within an agent-oriented system, called the Design Coordination System (DCS).

3. THE DCS

The DCS is aimed at the real-time operational coordination of a computational process. It incorporates the key elements of operational design coordination by encapsulating the appropriate techniques and managing the interrelationships between them. Section 3.1 gives an overview of the agent composition within the DCS. The architecture of the DCS is then presented in Section 3.2.

3.1. An overview of the DCS agent composition

The collection of agents operating within the DCS has been defined to satisfy the objective of conducting a computational process in an operationally coordinated manner. That is, the composition of agents, along with the role each fulfills, enables them to communicate with each other in real time such that they can perform activities involving task management, resource management, and schedule management simultaneously in a coherent manner. The behavior of all agents is complimentary in that they assist each other to satisfy the overall objective mentioned. More specifically, agents act as members of a cooperative, multifunctional team operating in a coordinated fashion to ensure that interdependent design tasks are completed in a structured manner with respect to time and to the allocation and utilization of the available resources. This process involves agents taking the opportunity to complete tasks concurrently when and where appropriate. However, the emphasis is placed on coordination, in that agent actions are performed appropriately with respect to the time and order that they are performed. As such, consistent with Lesser (1999), the collection of agents within the DCS can be described as a cooperative or benevolent agent society.

Before presenting an overview of the various types of agent, it is appropriate to define an analysis tool, a task, and a resource in the context of the DCS.

An analysis tool is a codified algorithm in the form of software that performs some numerical simulation (e.g., a computational fluid dynamics model). A task is a single execution of an analysis tool that uses data within some input files to create corresponding output files. Each execution of an analysis tool involves unique input files in terms of their name and contents. In addition, tasks are executions of analysis tools that, once started, must run to completion if they are to produce full and meaningful output.

A resource is an entity that is utilized to undertake tasks. In the context of the DCS, a resource is a workstation (e.g., a Sun Microsystems Ultra 10) in the computer network, on which analysis tools can be executed for given input.

The approach to operational design coordination involves tasks and resources being modeled appropriately, as shown in Tables A.1 and A.2 of Appendix A. A detailed description of the use of the knowledge attributes of tasks and resources is given in the industrial case study in Section 4, and a summary of the primary function of each agent type is presented in Table 1.

Table 1. *Primary functions of agent types*

Agent Type	Primary Function
Activity director	Implement schedules
Coordination manager	Facilitate communication links between related agents
Information manager	Manage input/output files related to analysis tool executions
Resource manager	Maintain knowledge of resources
Resource monitor	Sense, forecast, and disseminate resource performance efficiency
Scheduler	Perform scheduling
Task manager	Execute analysis tools given unique input data to create unique output data

Figure 1 indicates the location of the agent types within a computer network of workstations and the aspects of operational design coordination of each type of agent and communication links. In Figure 1, the computer network is shown as consisting of a workstation that is local to the user of the DCS and four remote workstations. The local workstation is that on which the user invokes the DCS. Remote workstations are used to perform executions of analysis tools.

With regard to Figure 1, and within any application of the DCS, a single coordination manager, scheduler and resource manager operate on the workstation local to the user of the system. Information managers are also situated on the local workstation, and one exists per analysis tool to be used in the computational process. A task manager is present for each analysis tool on every remote workstation being used in the computer network. Each remote workstation is also allocated a resource monitor and an activity director.

3.2. DCS architecture

As shown in Figure 2, the DCS comprises an agent framework, modeled knowledge, and user knowledge. The agent framework, which was reviewed in Section 3.1, acquires knowledge provided by the user to derive modeled knowledge. The agent framework then maintains and uses modeled knowledge through the application of real-time operational design coordination.

In Figure 2, a distinction is made between the communication between agents and the interactions between agents and knowledge modules, as the nature of these exchanges are different. That is, communication between agents involves asynchronous message passing using transmission control protocol/Internet protocol. Interactions between agents and knowledge modules entail agents extracting or modifying knowledge within the modules to enable them to

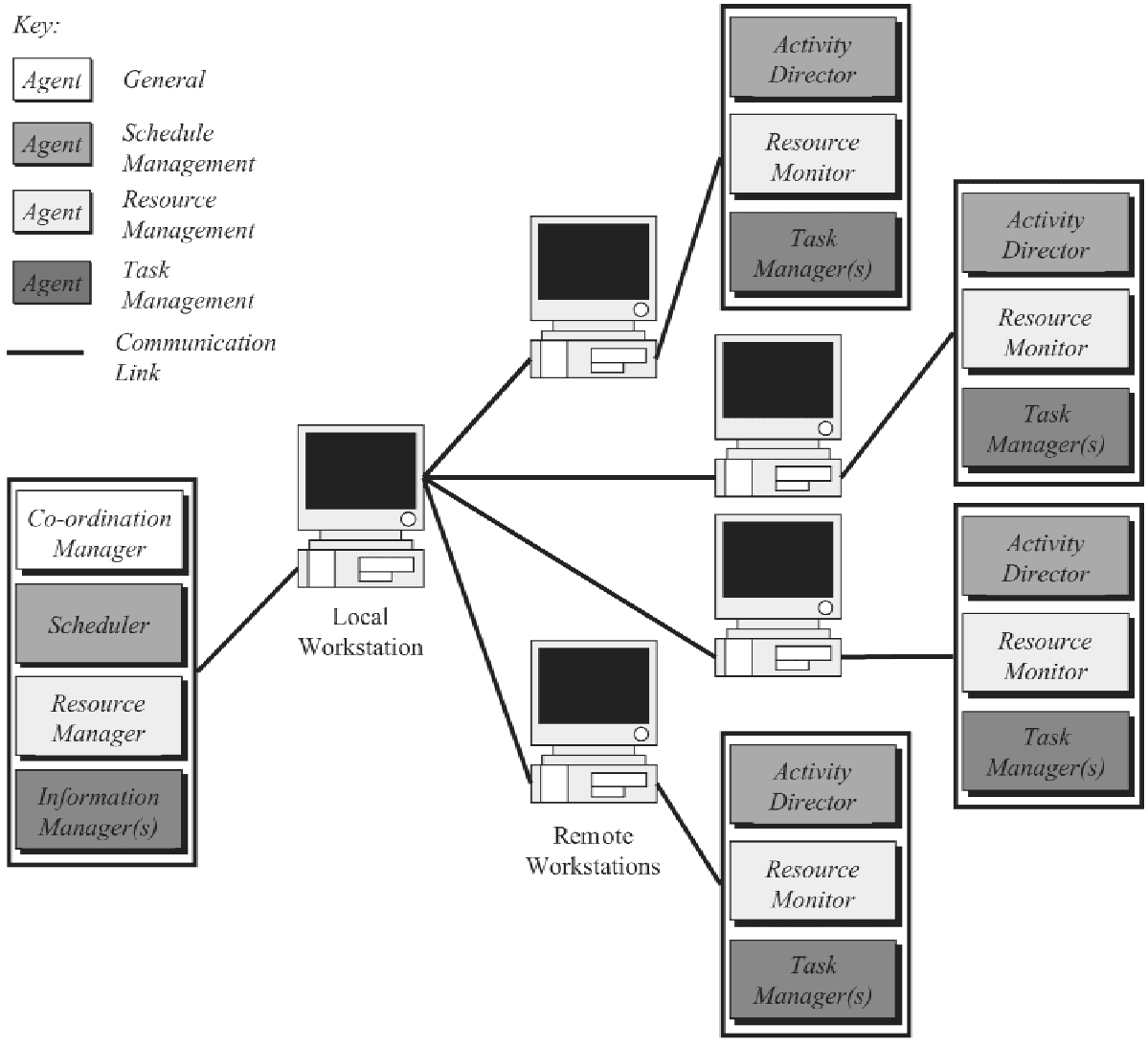

Fig. 1. The location of DCS agent types within a computer network of workstations.

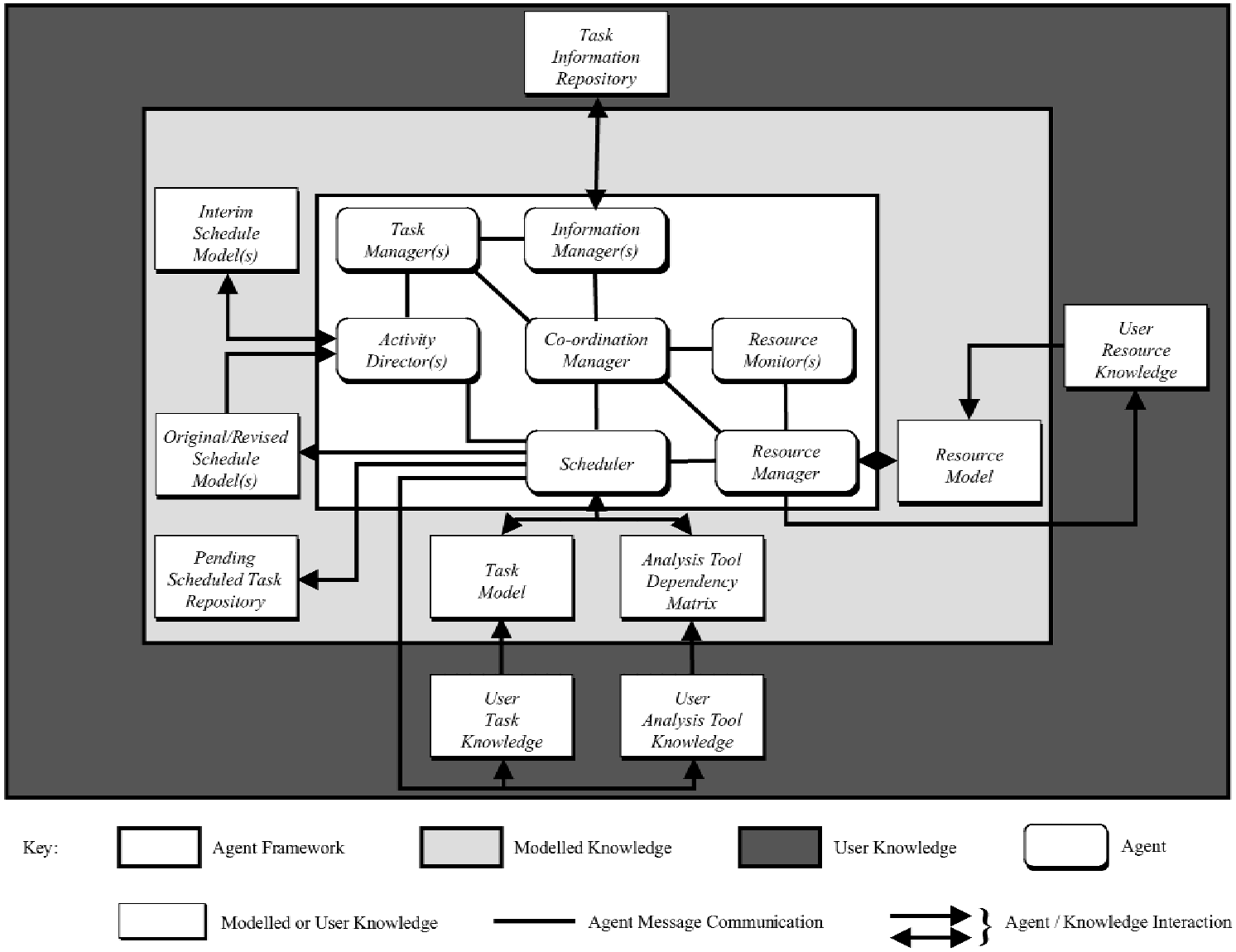

Fig. 2. The design coordination system architecture.

perform some action. These interactions are discussed throughout the industrial case study presented in Section 4.

With regard to Figure 2, a summary of the various modules associated with the modeled knowledge and user knowledge components of the DCS architecture is presented in Table 2.

The DCS is written in the C++ programming language and operated on a Unix platform consisting of a network of Sun Microsystems workstations (Ultra 1/170 and Ultra 10). Before using the DCS, its executable and those of all analysis tools must be copied into a directory on the local workstation, as illustrated in Figure 1. Once the DCS executable is invoked, knowledge must be provided by the user in accordance with Table 2 (i.e., knowledge of the analysis tools, tasks and associated information, and resources). That is, the user defines a profile for each analysis tool, which consists of knowledge of the input–output file requirements for each tool. On the basis of knowledge of the analysis tools, task knowledge is provided by the user in regard to the number of executions of each analysis tool. Any input file or files for the first analysis tool to be executed in the computational process are copied by the user and stored in the task information repository. The user also provides knowledge of the resources (i.e., host names of the remote workstations within the computer network that can be used by the DCS to execute analysis tools). On the basis of the knowledge provided by the user, the analysis tool dependency matrix, task model, and resource model are constructed and are then used and maintained by DCS agents throughout the computational process. The scheduler is the only agent with direct access to the analysis tool dependency matrix and task model. The resource model is only accessible via the resource manager.

4. ENGINEERING INDUSTRIAL CASE STUDY

Siemens Power Generation Limited provided a practical case study to enable the application of the DCS. Within the company, the Turbine Engineering Department is responsible for the design and development of turbine modules to upgrade/replace existing plants. The computational process of turbine blade design involves a suite of analysis tools that the designer uses in the selection of blades and blade path and in the calculation of the associated stresses and vibration characteristics of the blades. The deterministic analysis tools are related as shown in Figure 3. The naming convention of the analysis tools is company specific.

In Figure 3, the computational process is shown to involve 8 analysis tools. Further, analysis tool TF23225 is used for three purposes and, as such, is modeled as 3 individual analysis tools (i.e. TF23225_1, TF23225_2, and TF23225_3). Thus, the case study to be used consists of 10 analysis tools. For reasons of confidentiality, descriptions of these analysis tools were not divulged by the company. As a result, throughout this section, the analysis tools are referred to by their associated TF number.

Within Siemens Power Generation Limited, experienced design engineers manually manage the computational process of turbine blade design. That is, executions of analysis tools are performed sequentially, with the appropriate management of the large quantity of information and data held within files between each run. This means of managing information and data is time consuming and error prone. The manual enactment of a single run of the computational process, shown in Figure 3, takes approximately 8 min using a single workstation comparable to those used within the DCS (Whitfield et al., 2002). As such, the duration (min) of the complete computational process is approximately a factor of eight applied to the number of process runs. For comparative purposes of this case study, a single run of the computational process is considered.

4.1. Initialization

On instantiation of the DCS, a single coordination manager, resource manager, and scheduler are created. In this

Table 2. *Summary of modeled knowledge and user knowledge*

Component	Module	Description
Modeled knowledge	Analysis tool dependency matrix	Holds knowledge of the relationships between analysis tools
	Pending scheduled task repository	Holds knowledge of tasks that have been scheduled, which are awaiting other tasks to be completed
	Resource model	Holds knowledge of resources (i.e., workstations within the computer network)
	Schedule model	Holds knowledge of tasks that are to be undertaken on a specific resource (i.e., analysis tool executions for given input to be performed on specific workstations)
	Task model	Holds knowledge of tasks (i.e., analysis tool executions for unique input data)
User knowledge	Task information repository	Holds input files required to enable analysis tools to be executed and output files created as a result of executing analysis tools
	User analysis tool knowledge	Knowledge of the input/output file requirements of each analysis tool
	User resource knowledge	Knowledge of the workstations within the computer network
	User task knowledge	Knowledge of individual executions of analysis tools

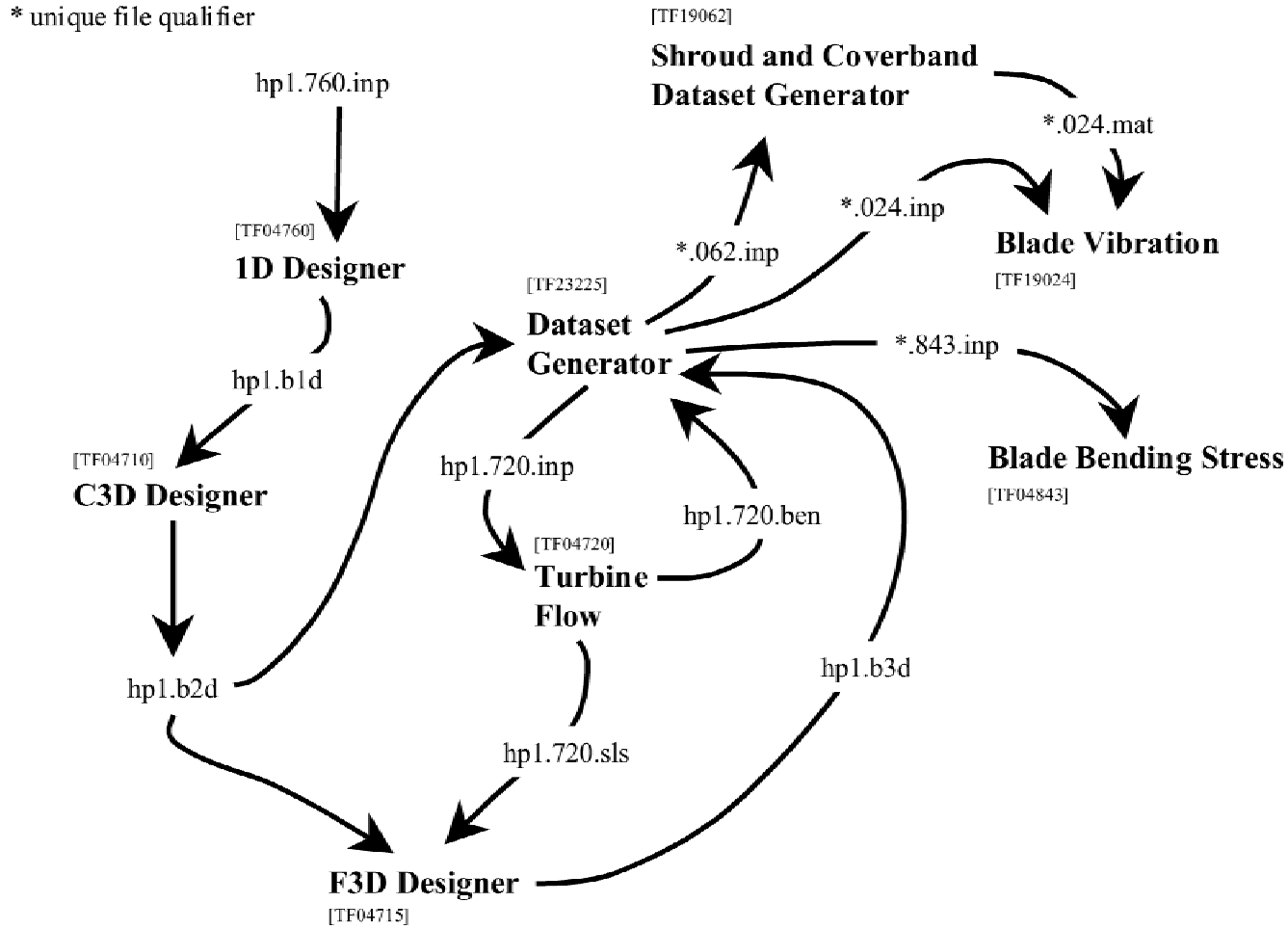

Fig. 3. The computational process of turbine blade design: analysis tools. *Unique file qualifier.

case study, 10 analysis tools and four resources are used. Thus, 10 information managers are also created, as well as 40 task managers, four resource monitors, and four activity directors.

Initially, the coordination manager receives messages from all agents. Knowledge contained within each initial communication relates to attributes of the agent, which is dependent on agent type. After recording these attributes, the manager replies to the agents, acknowledging their registration. Once registered, agents request knowledge of related agents from the coordination manager. This knowledge enables related agents to communicate directly with one another, via message passing, as and when required, such that they can work cooperatively and coordinate their actions.

4.1.1. Designer-defined tasks

The designer provides knowledge of the tasks to be undertaken (i.e., executions of analysis tools with unique input data). The computational process involves a number of executions of each of the analysis tools, as shown in Table 3, where T_I is a unique identification index for each analysis tool and n_{BR} is the number of turbine blade rows under consideration.

In this case study, the number of turbine blade rows under consideration is 36 (18 fixed and 18 rotating). Thus, the total number of tasks to be undertaken is 131.

Table 3. *Number of analysis tool executions*

T_I	Analysis Tool	No. of Executions
0	TF04760	1
1	TF04710	1
2	TF23225_1	1
3	TF04720	1
4	TF23225_2	n_{BR}
5	TF23225_3	$n_{BR}/2$
6	TF19062	$n_{BR}/2$
7	TF19024	$n_{BR}/2$
8	TF04843	n_{BR}
9	TF04715	1
	Total	$3.5\ n_{BR} + 5$

For each task, the designer allocates T_I, T_L, T_{DD}, $T_{[T_{In}]}$, and $T_{[T_{Out}]}$. The variable T_L is a task identification index that is local to its associated analysis tool. As an example, within the computational process, as analysis tool TF23225_2 (with $T_I = 4$) will be executed 36 times, each with a unique input file, then for $T_I = 4$, T_L ranges from 0 to 35. The variable T_{DD} is the datum duration of a task used for comparative purposes when considering resources of varying performance efficiency to undertake the task, and $T_{[T_{In}]}$ defines the information (i.e., input files) needed to be available before a task is undertaken. Similarly, $T_{[T_{Out}]}$ defines the information (i.e., output files) produced on the completion of a task.

4.1.2. Resource model

The construction and maintenance of the resource model is the primary responsibility of the resource manager. The upkeep of the knowledge held within the resource model provides the basis for the optimized utilization of the resources throughout the computational process.

The resource manager acquires user-supplied knowledge at the outset of the operation of the DCS and holds it in the resource model. Specifically, for each of the four resources to be used, the resource model consists of knowledge attributes as shown in Table 4. Here, R_I is a unique identification index, and R_A is an indication of whether a resource is available to be used, that is, $R_A = \{0, 1\}$, where 0 indicates that a resource is unavailable and 1 signifies that it is available. The variable R_{FE} is the forecasted performance efficiency expressed as a percentage. Within the operating environment of the DCS, R_{FE} is a measure of the potential performance efficiency that can be used and R_{LT} and R_{UT} are lower and upper performance efficiency thresholds, respectively, which if transgressed result in the consideration of rescheduling.

The resource manager maintains values of R_{FE} within the resource model with the assistance of the resource monitors. The initial values assigned to R_{FE} for each resource are calculated by the respective resource monitor based on values of monitored performance efficiency, R_{ME}, observed over a period of time before scheduling. The procedure of obtaining R_{FE} is explained in Sections 4.2.7 and 4.2.8.

4.1.3. Construct an analysis tool dependency matrix

An analysis tool dependency matrix is a representation of the relationships between the analyses tools involved in the computational process and is used to assist in the construction of a task model. To construct a dependency matrix, the scheduler uses knowledge provided by the user regarding each analysis tool to be used in the computational process. Knowledge of dependencies is established using the input and output requirements (i.e., the input and output files needed) for each analysis tool. By comparing the input requirements of each analysis tool, $T_{[T_{In}]}$, against the output requirements, $T_{[T_{Out}]}$, of all other analysis tools, the scheduler is able to determine the dependency relationships between them. Within the matrix, off-diagonal elements marked 0 represent nondependency and elements marked 1 signify dependency. The diagonal elements of the dependency matrix show the datum durations of executing each analysis tool, which are obtained from the user by performing arbitrary executions using a reference resource in the local area network. The dependency matrix for the computational process, shown in Figure 3, is presented in Table 5.

With regard to Table 5, as an example, for an execution of TF23225_2, the corresponding executions of TF04710 and TF04720 must be completed.

Table 4. *Resource model*

R_I	R_A	R_{FE} (%)	R_{LT} (%)	R_{UT} (%)
1	1	98.9	50	100
2	1	99.6	50	100
3	1	96.1	50	100
4	1	91.6	50	100

4.1.4. Construct a task model

A task model represents knowledge of the tasks to be undertaken in the computational process (i.e., executions of the analysis tools and the unique files required to do so). The scheduler is responsible for ensuring that the task model is constructed and maintained throughout the computational process. To construct a task model, the scheduler uses knowledge provided by the user and contained within the analysis tool dependency matrix.

Table 6 represents the analysis tool dependency matrix shown in Table 5, with each analysis tool assigned a value for T_I in accordance with Table 3. In addition, Table 7 represents some of the knowledge of tasks held within the task model. For the analysis tool with $T_I = 6$ (i.e., TF19062), two tasks are shown, each with different values assigned to T_L because they have unique input and output files signified by $T_{[T_{In}]}$ and $T_{[T_{Out}]}$. Using the analysis tool dependency matrix, it can be determined that input for each task associated with the analysis tool with $T_I = 6$ is created as output from the analysis tool with $T_I = 5$.

As stated in Section 4.1.1, knowledge of tasks defined by the designer comprises T_I, T_L, T_{DD}, $T_{[T_{In}]}$, and $T_{[T_{Out}]}$. Within the task model, tasks are assigned additional knowledge (namely, T_G, T_C, T_N, and $T_{[T_G]}$). The variable T_G is an identification index for a task within the context of all tasks and is used for scheduling and rescheduling purposes. With regard to the computational process, because there are a total of 131 analysis tool executions, the T_G value ranges from 0 to 130. The variable T_C represents whether or not a task has been completed such that $T_C = \{0, 1\}$, where 0 indicates noncompletion and 1 signifies completion. In addition, the analysis tool dependency matrix, which was constructed using knowledge of $T_{[T_{In}]}$ and $T_{[T_{Out}]}$ for each analysis tool, is used to determine T_N and $T_{[T_G]}$ for each task. The variable T_N is the

Table 5. *Analysis tool dependency matrix*

	TF04760	TF04710	TF23225_1	TF04720	TF23225_2	TF23225_3	TF19062	TF19024	TF04843	TF04715
TF04760	97	0	0	0	0	0	0	0	0	0
TF04710	1	6	0	0	0	0	0	0	0	0
TF23225_1	0	1	6	0	0	0	0	0	0	0
TF04720	0	0	1	14	0	0	0	0	0	0
TF23225_2	0	1	0	1	1	0	0	0	0	0
TF23225_3	0	1	0	0	0	2	0	0	0	0
TF19062	0	0	0	0	0	1	1	0	0	0
TF19024	0	0	0	0	0	1	1	1	0	0
TF04843	0	0	0	1	1	0	0	0	1	0
TF04715	0	1	0	1	0	0	0	0	0	11

number of dependencies of a task, and $T_{[T_G]}$ is a matrix defining T_G of each of these dependencies.

4.2. Operation

For the purposes of this case study, the operation of the agents within the DCS has been divided into subsections 4.2.1 to 4.2.16. The order in which these subsections are presented corresponds to the occurrence of agent actions during the computational process.

4.2.1. Derive an original schedule

The primary role of the scheduler is to satisfy the objectives of scheduling a number of tasks on the least number of available resources while consuming the least amount of those resources, such that dependencies are preserved and the overall time to complete the tasks is minimized. At the outset of the computational process, the scheduler derives an original schedule using a multiobjective genetic algorithm (MOGA; Todd, 1997; Todd & Sen, 1997*a*, 1997*b*) allied with knowledge of outstanding tasks and available resources from their respective models. Initially, all tasks are outstanding (i.e., for each task $T_C = 0$) and, thus, all need to be scheduled. Task knowledge required for use with the MOGA comprises T_G, T_{DD}, T_N, and $T_{[T_G]}$ for each task. Furthermore, the scheduler notes the number of tasks to be scheduled (n_{TS}) and the cumulative number of dependencies for those tasks (n_{TD}). As indicated in the resource model

Table 6. *Analysis tool dependency matrix*

T_I	0	1	2	3	4	5	6	7	8	9
0	97	0	0	0	0	0	0	0	0	0
1	1	6	0	0	0	0	0	0	0	0
2	0	1	6	0	0	0	0	0	0	0
3	0	0	1	14	0	0	0	0	0	0
4	0	1	0	1	1	0	0	0	0	0
5	0	1	0	0	0	2	0	0	0	0
6	0	0	0	0	0	1	1	0	0	0
7	0	0	0	0	0	1	1	1	0	0
8	0	0	0	1	1	0	0	0	1	0
9	0	1	0	1	0	0	0	0	0	11

Table 7. *Analysis tool task model*

T_I	T_L	…	$T_{[T_{In}]}$	$T_{[T_{Out}]}$
0	0	…	hp1.760.inp	hp1.720.inp
1	0	…	hp1.b1d	hp1.b1d
2	0	…	hp1.b2d	hp1.b2d
3	0	…	hp1.720.inp	hp1.720.sls, hp1.720.ben
…	…	…	…	…
6	12	…	hp1.026.062.inp	hp1.026.062.out, hp1.026.024.mat
6	13	…	hp1.028.062.inp	hp1.028.062.out, hp1.028.024.mat
…	…	…	…	…
…	…	…	…	…
9	0	…	hp1.b2d, hp1.720.ben	hp1.b3d

shown in Table 4, each of the four resources are available for utilization because $R_A = 1$ for all of them. With regard to each of the available resources, the scheduler requests that the resource manager provide R_I and R_{FE}. The scheduler also notes the number of resources to be considered for scheduling, n_{RS}.

Executing the MOGA results in a set of solutions (i.e., schedules) being created that satisfy the three objectives defined above. Figure 4 illustrates a set of solutions created by using the MOGA with respect to minimizing the conflicting objectives.

Within the set of solutions created, a subset exists known as the Pareto (1896) optimal set of solutions (i.e., schedules). A Pareto optimal set comprises solutions in which no increase can be achieved in any of the criteria without resulting in a simultaneous decrease in at least one of the remaining criteria.

The scheduler identifies the Pareto optimal set of solutions and then uses multicriteria decision making to select the most appropriate, or best, schedule from this set. The criteria used coincide with the objectives of completing the computational process in the least time using the least number of resources while consuming the least amount of those resources. That is, they strive to minimize time, number of resources, and resource use. The first criterion applied to the Pareto optimal set is that of minimizing the time estimate to complete all scheduled tasks. Thus, the schedule with the least time estimate to complete the tasks scheduled is selected. In the event that more than one schedule within the Pareto optimal set satisfies this first criterion, a second criterion is applied such that the schedule with the least number of resources used to achieve the completion of the tasks scheduled is chosen. Again, in the event that more than one schedule satisfies the first and second criterion, a third, and final, criterion is used. That is, the schedule exhibiting the least cumulative percentage utilization of the resources employed is selected. If there are a number of schedules within the Pareto optimal set that satisfy all criteria, then a schedule is arbitrarily chosen because all of these schedules are equally as good.

An example original schedule selected from the Pareto optimal set of solutions is shown in Figure 5.

4.2.2. Construct original schedule models

The scheduler uses the best optimized schedule to construct an original schedule model for each resource to be used. The responsibility of administering the enactment of each original schedule model lies with the activity director of the corresponding resource. Thus, the scheduler notifies and provides each activity director with the respective original schedule model. In Table 8, the best schedule is presented in the form of an original schedule model for resource

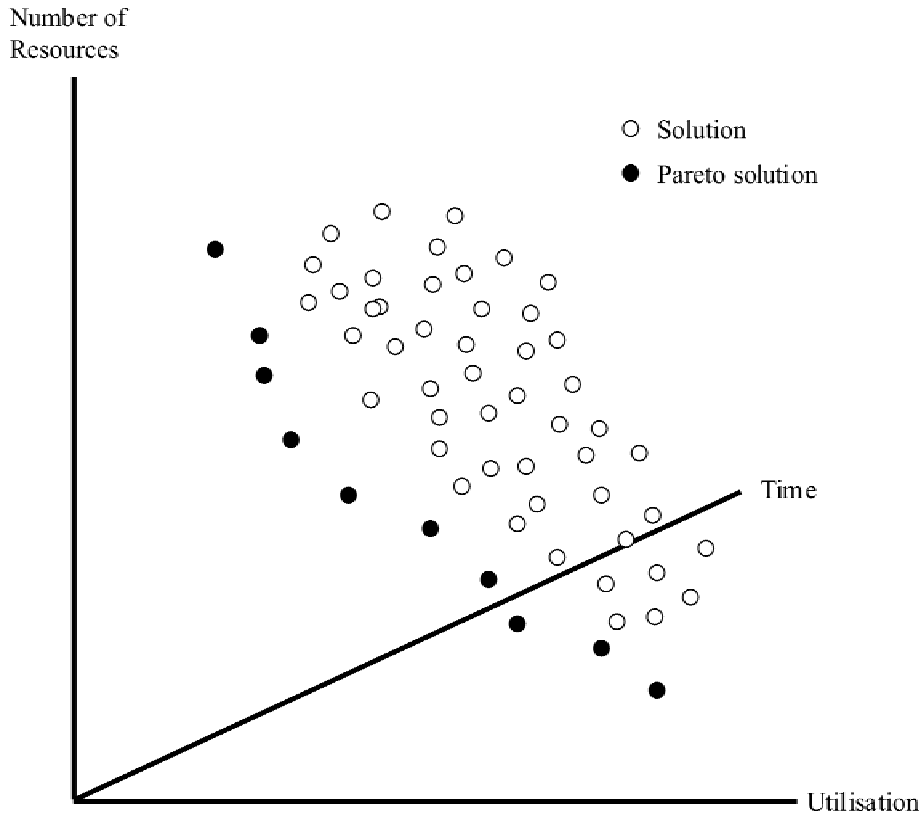

Fig. 4. The trade-off between conflicting objectives.

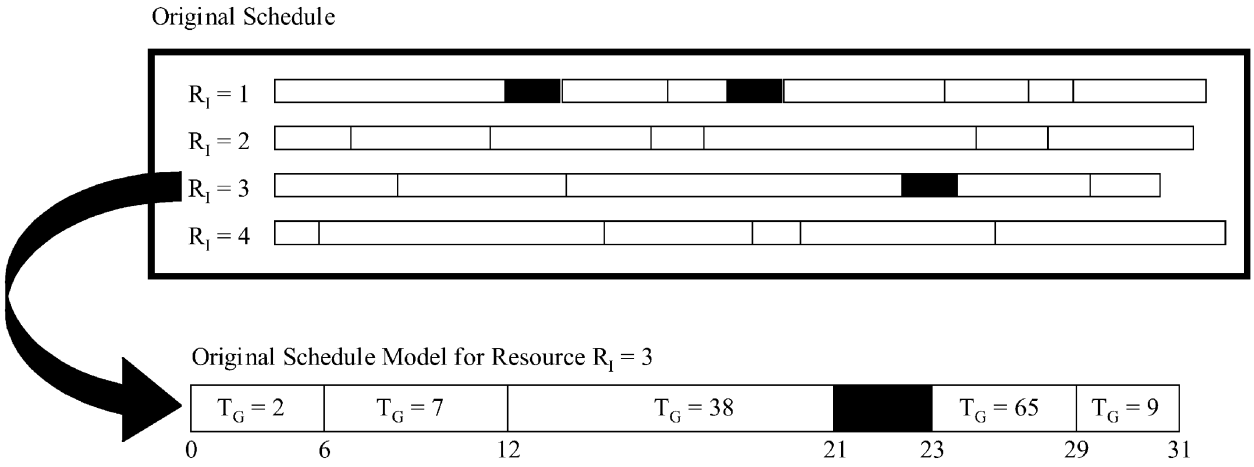

Fig. 5. An optimized schedule.

$R_I = 1$. Furthermore, tasks are listed in the order that they should be undertaken.

In Table 8 $T_{[T_I]}$ and $T_{[T_L]}$ are matrices defining T_I and T_L of each dependency of a task.

4.2.3. Check dependencies and direct tasks

On being provided their original schedule models, each activity director begins administering the designated tasks to be undertaken. As indicated in Table 8, the task with $T_I = 1$ and $T_L = 0$ is the first task to be undertaken using resource $R_I = 1$ (i.e., the only execution of analysis tool TF04710 using input file hp1.b1d and creating output file hp1.b2d). For brevity, in the remainder of this article, a task's T_I and T_L will be abbreviated to $T(T_I, T_L)$; that is, the task previously stated is denoted as $T(1, 0)$.

On inspecting its associated original schedule model, the activity director recognizes that this task is dependent on the completion of $T(0, 0)$; that is, the single execution of analysis tool TF04760 as shown in Figure 6. This dependency exists because TF04710 requires the output file produced on executing TF04760 (i.e., hp1.b1d) as input. Thus, the activity director confers with the scheduler to establish whether the dependency has been completed. The scheduler checks the task model to determine T_C of the dependency $T(0, 0)$.

Because the execution of TF04760 has not been completed (i.e., $T_C = 0$), the execution of TF04710 cannot commence. As such, the scheduler records within the pending scheduled task repository that $T(1, 0)$ is awaiting the completion of $T(0, 0)$. Similarly, as shown in Figure 6, the first task to be undertaken using resource $R_I = 3$ [i.e., $T(5, 4)$] and $R_I = 4$ [i.e., $T(5, 11)$] cannot start because it requires information that will only become available on the completion of $T(1, 0)$. At the outset of the computational process, only $T(0, 0)$ on resource $R_I = 2$ is able to commence, because it has no dependencies. The pending scheduled task repository at the point described is shown in Table 9, where $i = \{1, 2, \ldots, n_{PST}\}$, $j = \{1, 2, \ldots, T_{ON,i}\}$, n_{PST} is the number of

Table 8. *Original schedule model for resource* $R_I = 1$

T_I	T_L	T_G	T_{DD}	T_N	$T_{[T_I]}$, $T_{[T_L]}$	T_I	T_L	T_G	T_{DD}	T_N	$T_{[T_I]}$, $T_{[T_L]}$
1	0	1	6	1	[0], [0]	4	5	9	1	2	[1, 3], [0, 0]
2	0	2	6	1	[1], [0]	8	20	114	1	2	[3, 4], [0, 20]
5	7	47	2	1	[1], [0]	4	0	4	1	2	[1, 3], [0, 0]
5	0	40	2	1	[1], [0]	7	9	85	1	2	[5, 6], [9, 9]
5	8	48	2	1	[1], [0]	8	18	112	1	2	[3, 4], [0, 18]
6	12	70	1	1	[5], [12]	4	3	7	1	2	[1, 3], [0, 0]
7	15	91	1	2	[5, 6], [15, 15]	8	9	103	1	2	[3, 4], [0, 9]
5	10	50	2	1	[1], [0]	8	21	115	1	2	[3, 4], [0, 21]
5	16	56	2	1	[1], [0]	4	26	30	1	2	[1, 3], [0, 0]
6	14	72	1	1	[5], [14]	4	32	36	1	2	[1, 3], [0, 0]
4	33	37	1	2	[1, 3], [0, 0]	8	19	113	1	2	[3, 4], [0, 19]
9	0	130	11	2	[1, 3], [0, 0]	8	13	107	1	2	[3, 4], [0, 13]
4	13	17	1	2	[1, 3], [0, 0]	8	12	106	1	2	[3, 4], [0, 12]

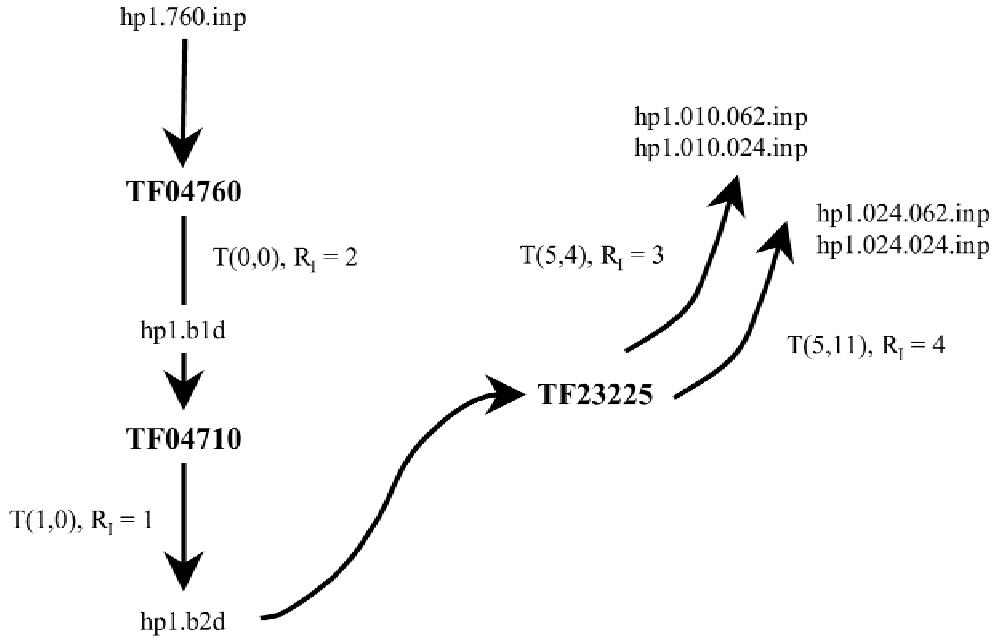

Fig. 6. The initial tasks to be undertaken.

pending scheduled tasks, and T_{ON} is the number of outstanding dependencies of a pending scheduled task.

In the event of a task being ready to be undertaken that has no dependencies, the activity director omits this consultation with the scheduler and directly contacts the relevant task manager, instructing it to commence undertaking the specified task.

4.2.4. Request, provide, and supply task information/undertake tasks

The principal duty of a task manager is the execution of its associated analysis tool for a unique input file or files. An instruction to a task manager indicating that a specific task should commence is provided by its related activity director. As such, before undertaking the task $T(0, 0)$, that is, executing analysis tool TF04760, the relevant task manager requests that its related information manager provide the necessary input file according to $T_{[T_{In}]}$ (i.e., hp1.760.inp). In response, the information manager locates and retrieves this input file from the task information repository. On notification that the requested information has been provided, the task manager commences with the execution of its associated analysis tool. Once TF04760 has been executed, the task manager informs its related information manager such that the output file created (i.e., hp1.b1d, in accordance with $T_{[T_{Out}]}$), can be stored in the task information repository, which is accessible by all information managers. Thus, these files are available in the event of them being required as input for the execution of other analysis tools, specifically, at this time in the process, the pending scheduled task $T(1, 0)$.

Table 9. *Pending scheduled task repository*

i	T_I	T_L	T_{ON}	$[T_{I,i,j}]$	$[T_{L,i,j}]$
1	1	0	1	0	0
2	5	4	1	1	0
3	5	11	1	1	0

4.2.5. Update task model

On completion of $T(0, 0)$, the task manager informs its related activity director, which then informs the scheduler of this fact. The scheduler updates the task model to reflect the completion of the task by setting $T_C = 1$. Updating the task model ensures that in the event of rescheduling, only outstanding tasks will be considered (i.e., tasks with $T_C = 0$).

4.2.6. Remove dependencies and commence direction of pending scheduled tasks

In addition to updating the task model, the scheduler updates the pending scheduled task repository. As such, any tasks solely awaiting the completion of the recently completed dependency may be undertaken. Specifically, as $T(0, 0)$ has been completed, the scheduler removes this dependency from the pending scheduled task repository and decrements T_{ON} where appropriate. As a consequence, as T_{ON} becomes nil for $T(1, 0)$, that is, this task has no outstanding dependencies, it can be undertaken. The commencement of this task is instigated by the scheduler, who informs the appropriate activity director.

4.2.7. Monitor resources

Each resource monitor is responsible for sensing, forecasting, and reporting performance efficiency of its associated resource. Throughout the computational process, each resource monitor observes the various constituents of central processing unit (CPU) utilization of its associated resource such that any violations of the upper or lower performance efficiency thresholds can be identified. In particular, a resource monitor establishes what percentage of the current CPU utilization of its associated resource is attributed to

- user processes, R_{user}, which are the computer programs being run by users;
- system processes, R_{system}, which is UNIX kernel code; and
- idle, R_{idle}, which is not being utilized.

Furthermore, R_{user} is divided into the proportion of CPU utilization attributed to computer programs that are being executed within the DCS (i.e., analysis tools, R_{DCS}) and that are unrelated to the DCS (R_{other}). On the basis of observations of CPU utilization over a period of time, each resource monitor calculates the monitored performance efficiency of its associated resource at time t, R_{ME_t}, using the equation

$$R_{\text{ME}_t} = R_{\text{CF}} \times \left[R_{\text{idle}_t} + R_{\text{DCS}_t} + \frac{R_{\text{other}_t}}{n_{\text{ps}}} (1 + R_{\text{system}_t}) \right].$$

The resource coefficient (R_{CF}) is a relative measure of the processor speed of a resource such that the forecasted performance efficiencies determined for all resources are directly comparable. This is required for purposes of scheduling and rescheduling. In addition, n_{ps} is the number of processes being executed on the resource.

The CPU utilization and monitored performance efficiency of resource $R_{\text{I}} = 4$ over a period of the computational process are shown in Figure 7 as observed by the associated resource monitor.

In Figure 7, it can be seen that at approximately $t = 60$ s there is a deviation in monitored performance efficiency that transgresses its lower threshold of $R_{\text{LT}} = 50\%$, which instigates the consideration of rescheduling. Further, for this resource, R_{ME} fluctuates between 38 and 48% during the remainder of the computational process. Although not shown in Figure 7, for resources $R_{\text{I}} = 1$, 2, and 3, monitored performance efficiency is approximately 99% for the remainder of the computational process.

4.2.8. Forecast and revise resource model

Because of the monitored performance efficiency of resource $R_{\text{I}} = 4$ falling below the lower threshold of 50%,

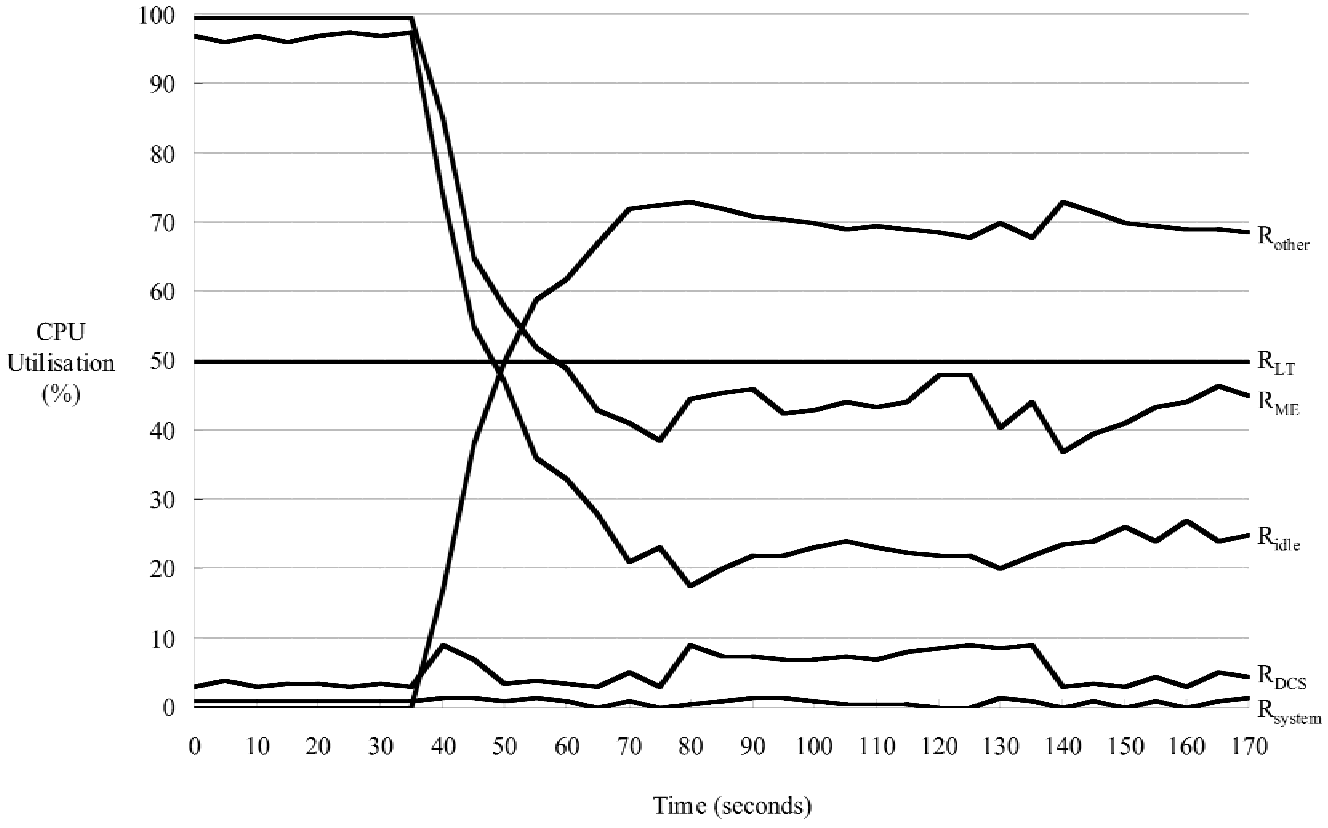

Fig. 7. The constituent usage and monitored performance efficiency versus time for $R_{\text{I}} = 4$.

the associated resource monitor forecasts future performance efficiency. This is achieved by performing a regression analysis using orthogonal polynomials with recent values of monitored performance efficiency. The regression equation derived for resource $R_I = 4$ is

$$R_{FE,\,t} = -0.1979t^4 + 7.7977t^3 - 111.97t^2 + 683.18t - 1395.9.$$

The resource monitor supplies the forecasted performance efficiency to the resource manager, which updates the resource model accordingly. The resource manager also resets the values of R_{LT} or R_{UT} to 10 and 70%, respectively. The reason for resetting these values is to avoid rescheduling being considered again as a result of insignificant deviations in monitored performance efficiency about the previous lower threshold. The resource manager also requests that all other resource monitors determine and report forecasts of performance efficiency for their associated resource. The resource manager then updates the resource model, as shown in Table 10.

Subsequently, based on the up to date knowledge of resource forecasted performance efficiencies, the resource manager instructs the scheduler to consider rescheduling.

4.2.9. Decision making for rescheduling/deriving interim schedule models

The decision-making process for rescheduling involves the scheduler assessing whether it would be more economical timewise to continue adhering to the current schedule or to derive and enact a revised schedule. If the time taken to complete the current schedule is greater than the time taken to derive and complete a revised schedule, then the decision is made to reschedule. Otherwise, the current schedule is continued.

If rescheduling a proportion of the outstanding tasks is performed, during this period the remaining outstanding tasks can be completed in accordance with interim schedule models. These models are derived as a by-product of estimating the time to derive a revised schedule. The concept of interim schedule models ensures that if rescheduling does occur, then the resources remain utilized appropriately during this period.

Before calculating the time estimates mentioned above, the scheduler requests that all activity directors suspend their original/revised schedule models. This ensures that knowledge of tasks considered by the scheduler is not subject to change during the decision-making process. Determining the time needed to complete the current schedule, derive a revised schedule, and enact a revised schedule involves the consideration of the 104 outstanding tasks within the original schedule models of each activity director, as presented in Table 11. Shaded cells in Table 11 signify those outstanding tasks that could potentially be included within an interim schedule model because they are independent (i.e., $T_N = 0$) at the point at which the scheduler considers rescheduling. Further, the estimated durations of tasks (T_{ED}) are summed to determine the cumulative time required to complete the tasks that could potentially be included in the interim schedule models.

Table 10. *Revised resource model*

R_I	R_A	R_{FE} (%)	R_{LT} (%)	R_{UT} (%)
1	1	99.1	50	100
2	1	99.2	50	100
3	1	99.0	50	100
4	1	41.8	10	70

4.2.10. Estimated time to complete the current schedule

Estimating the time to complete the current (i.e., original) schedule (T_{CCS}) initially involves the scheduler supplying up to date resource forecasted performance efficiency to each activity director. The activity directors then apply the forecasted performance efficiency of their associated resource to the cumulative datum duration of the outstanding tasks within their original schedule models, as shown in Table 12.

Each activity director provides the scheduler with this estimation of the time to complete the associated original schedule model. The scheduler then determines that the original schedule model with the greatest estimated completion time, indicated by the shaded cells, corresponds with resource $R_I = 4$. That is, the resource that experienced the significant reduction in forecasted performance efficiency from 91.6 to 41.8%. Thus, if the original schedule models continue to be adhered to under the prevailing forecasted performance efficiency, it is estimated that they would be completed in approximately 79 s (i.e., $T_{CCS} = 79$ s).

4.2.11. Estimated time to derive a revised schedule

To estimate the time to derive a revised schedule (T_{DR}), the scheduler uses empirically derived knowledge of the execution time of the MOGA and knowledge of the outstanding tasks within the original schedule models. Figure 8 presents the empirical relationship between the number of tasks to be scheduled (n_{TS}) for a number of resources to be utilized (n_R) and the estimated execution time of the MOGA (T_{MOGA}). The number of tasks to be scheduled ranges from 20 to 127 because using the MOGA to reschedule beyond these limits would be uneconomical in terms of time.

Estimating the time to derive a revised schedule involves establishing the most appropriate combination of outstanding tasks to reschedule, whereas the remainder can be completed during this period such that the most appropriate utilization of resources is maintained. Determining the appropriate combination of tasks involves the application of an

Table 11. *Outstanding tasks within original schedule models*

$R_I = 1$, $R_{FE} = 99.1\%$				$R_I = 2$, $R_{FE} = 99.2\%$				$R_I = 3$, $R_{FE} = 99.0\%$				$R_I = 4$, $R_{FE} = 41.8\%$			
T_G	T_N	T_{DD}	ΣT_{ED}	T_G	T_N	T_{DD}	ΣT_{ED}	T_G	T_N	T_{DD}	ΣT_{ED}	T_G	T_N	T_{DD}	ΣT_{ED}
72	1	1	—	76	1	1	—	28	0	1	1.01	65	0	1	2.39
37	0	1	1.01	43	0	2	2.02	86	1	1	—	83	1	1	—
130	0	11	12.11	88	0	1	3.02	74	0	1	2.02	58	0	1	4.78
17	0	1	13.12	60	0	1	4.03	34	0	1	3.03	49	0	2	9.57
9	0	1	14.13	54	0	2	6.05	64	0	1	4.04	81	0	1	11.96
114	1	1	—	53	0	2	8.07	12	0	1	5.05	68	0	1	14.35
4	0	1	15.14	24	0	1	9.07	102	1	1	—	78	1	1	—
85	2	1	—	35	0	1	10.08	39	0	1	6.06	16	0	1	16.75
112	1	1	—	33	0	1	11.09	87	0	1	7.07	118	1	1	—
7	0	1	16.15	18	0	1	12.10	21	0	1	8.08	25	0	1	19.14
103	1	1	—	19	0	1	13.11	66	0	1	9.09	92	1	1	—
115	1	1	—	109	1	1	—	31	0	1	10.10	123	1	1	—
30	0	1	17.15	22	0	1	14.11	117	1	1	—	8	0	1	21.53
36	0	1	18.16	90	2	1	—	67	1	1	—	26	0	1	23.92
113	1	1	—	108	1	1	—	15	0	1	11.11	116	1	1	—
107	1	1	—	38	0	1	15.12	105	1	1	—	129	1	1	—
106	1	1	—	128	1	1	—	94	1	1	—	14	0	1	26.32
				111	1	1	—	96	1	1	—	104	1	1	—
				121	1	1	—	59	0	1	12.12	27	0	1	28.71
				100	1	1	—	97	1	1	—	10	0	1	31.10
				99	1	1	—	61	1	1	—	32	0	1	33.49
				6	0	1	16.13	79	1	1	—	71	1	1	—
				5	0	1	17.14	23	0	1	13.13	98	1	1	—
				122	1	1	—					11	0	1	35.89
				13	0	1	18.15					101	1	1	—
				89	2	1	—					95	1	1	—
				82	1	1	—					84	1	1	—
				20	0	1	19.15					29	0	1	38.28
				120	1	1	—					127	1	1	—
				77	1	1	—					110	1	1	—
				124	1	1	—					126	1	1	—
				119	1	1	—					125	1	1	—

iterative procedure. Step 1 involves estimating the execution time of the MOGA given the number of tasks to be rescheduled and the number of resources to be utilized. On the basis of the time estimate from step 1, step 2 entails using the original schedule model for each resource to ascertain the number of outstanding tasks that could be completed during rescheduling. Step 3 involves deducting the cumulative number of outstanding tasks able to be completed utilizing all resources determined in step 2 from the number of tasks considered for rescheduling in step 1. The results from applying the procedure are shown in Table 13, where n_{TRS} is the number of outstanding tasks to reschedule, n_{TCRS} is the number of tasks that could be completed using a particular resource during rescheduling given the estimated time to derive a revised schedule using the MOGA, and T_{TCRS} is the time taken to complete a number of tasks using a particular resource during rescheduling.

Table 12. *Estimated times to complete current schedule models*

R_I	ΣT_{DD} (s)	R_{FE} (%)	$\Sigma T_{ED} = \Sigma T_{DD}/R_{FE}$ (s)
1	27	99.1	27.2
2	35	99.2	35.3
3	23	99.0	23.2
4	33	41.8	78.9

The procedure converges when the time taken to reschedule a number of outstanding tasks is as nearly coincident as possible with the completion of those remaining. Thus, the idle time of each resource is minimized. Table 13 shows that convergence to the optimum solution with respect to concurrent rescheduling/task completion is reached after four iterations. That is, the scheduler should reschedule 60 tasks, estimated to take approximately 20 s (i.e., T_{DRS} = 20.3 s), according to the regression equation associated with four resources shown in Figure 8. During the period of rescheduling, 44 tasks are estimated as being able to be

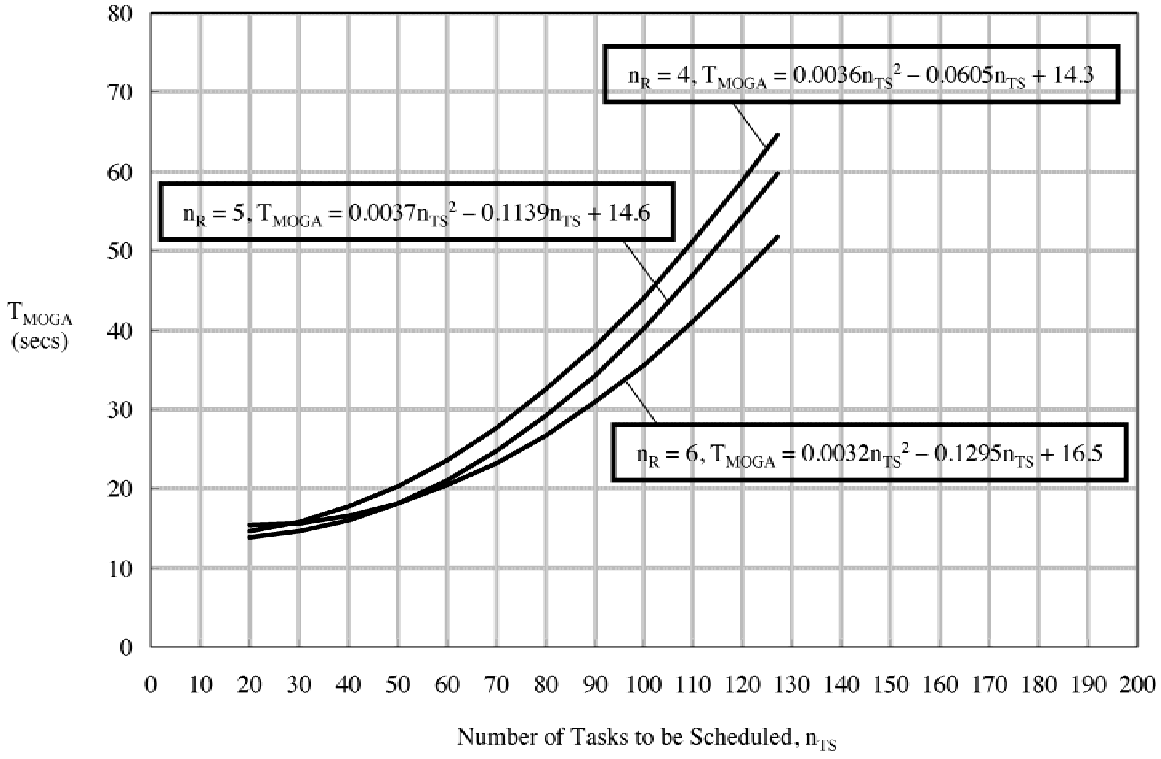

Fig. 8. The estimated execution time of MOGA.

completed using the four resources. On the basis of their most recent forecasted performance efficiency, resources $R_I = 1$, 2, and 4 would be utilized for approximately 19 s, whereas resource $R_I = 3$ for approximately 13 s. As a consequence, not only has the optimized time to reschedule an appropriate number of outstanding tasks been determined but also the tasks to be completed during this period have been identified (i.e., those for inclusion within the interim schedule models). Values of T_G within the interim schedule models are shown in Table 14.

Concurrent rescheduling/task completion results in a mean resource idle time of approximately 3 s. Because resources idle time is minimized, the arrival of the revised schedule is expected to be as close as possible to the completion of the interim schedule models.

4.2.12. Estimated time to complete a revised schedule

Estimating the time to complete a revised schedule (T_{CRS}) involves simulating the grouping and assignment of tasks to be rescheduled to the allocated resources. Grouping tasks enables the identification of groups that must be undertaken sequentially, which consist of tasks that may be completed concurrently. Assigning tasks to resources is done in accordance with these groups such that the greatest cumulative time to complete the tasks can be obtained.

Table 13. *Determination of time to reschedule and concurrently complete tasks*

			Resources								
			$R_I = 1$		$R_I = 2$		$R_I = 3$		$R_I = 4$		
Iteration	n_{TRS}	T_{MOGA} (s)	n_{TCRS}	T_{TCRS} (s)	n_{TCRS}	T_{TCRS} (s)	n_{TCRS}	T_{TCRS} (s)	n_{TCRS}	T_{TCRS} (s)	Σn_{TCRS}
1	104	37.6	8	18.16	16	19.15	13	13.13	14	35.89	51
2	53	18.6	8	18.16	15	18.15	13	13.13	6	16.75	42
3	62	20.8	8	18.16	16	19.15	13	13.13	7	19.14	44
4	60	20.3	8	18.16	16	19.15	13	13.13	7	19.14	44

Table 14. *Interim schedule models*

R_I	T_G															
1	37	130	17	9	4	7	30	36								
2	43	88	60	54	53	24	35	33	18	19	22	38	6	5	13	20
3	28	74	34	64	12	39	87	21	66	31	15	59	23			
4	65	58	49	81	68	16	25									

STEP 1. *Determine task groups:* To determine the groups into which the 60 tasks to be rescheduled could be divided, an assessment of the tasks dependencies is made (i.e., whether it/they was/were completed in accordance with an original schedule model, will be completed in accordance with an interim schedule model, or will be rescheduled for inclusion with a revised schedule model).

Forty-eight tasks would not have any outstanding dependencies once rescheduled because

- they never had any dependencies,
- their dependencies were completed in accordance with the original schedule models before the consideration of rescheduling, or
- their dependencies will be completed in accordance with the interim schedule models during the period of rescheduling.

Similarly, as a result of rescheduling, only 12 tasks will have outstanding dependencies within the revised schedule models. As a consequence, the 60 tasks to be considered for rescheduling can be divided into two groups, the first group comprising 48 tasks and the second group consisting of 12 tasks. Further, these groups must be completed sequentially. However, tasks within each group may be completed in parallel because they are independent of other tasks in the same group.

STEP 2. *Distribute group tasks among resources:* Given that the datum duration of each outstanding task to be rescheduled is 1 s, Table 15 presents information regarding how the groups of tasks identified in Step 1 could be distributed among the four resources such that their collective time to completion is minimized.

Based on Table 15, the T_{CRS} is approximately 18 s. This corresponds to the greatest cumulative time to complete the groups of tasks, as indicated by the shaded row of Table 15.

4.2.13. Decision to reschedule

The scheduler makes the decision to reschedule because the estimated time to complete the original schedule is greater than the time it is estimated to derive and complete a revised schedule, which is

$$T_{CCS}\ (79\ \text{s}) > T_{DRS}\ (20\ \text{s}) + T_{CRS}\ (18\ \text{s}).$$

Furthermore, the scheduler instructs each activity director to administer the interim schedule model during the period of rescheduling. These models (Table 14) were constructed as a by-product of determining the estimated time to derive a revised schedule. In addition, the completion of the interim schedule models is intended to be near coincident with the completion of rescheduling. Thus, the transition delay between the current and revised schedules is minimized.

4.2.14. Modify task model

Before rescheduling, the scheduler modifies the task model. This is required because knowledge held in the task model is used in the derivation of the revised schedule models. Knowledge in the task model modified by the scheduler consists of the T_C of those tasks to be completed in accordance with the interim schedule models, the T_G, $T_{[T_G]}$, and

Table 15. *Assignment of rescheduled tasks*

		Group 1: 48 Tasks		Group 2: 12 Tasks		
R_I	R_{FE} (%)	No. of Tasks Assigned	$\Sigma T_{ED} = \Sigma T_{DD}/R_{FE}$ (s)	No. of Tasks Assigned	$\Sigma T_{ED} = \Sigma T_{DD}/R_{FE}$ (s)	Total Time (s)
1	99.1	14	14.13	4	4.04	18.17
2	99.2	14	14.11	4	4.03	18.14
3	99.0	14	14.14	3	3.03	17.17
4	41.8	6	14.35	1	2.39	16.74

T_N of the 60 outstanding tasks to be rescheduled and considered in Table 15.

4.2.15. Derive revised schedule models/complete interim schedule models

To derive the revised schedule models, the scheduler uses the MOGA with knowledge held within the modified task model and revised resource model. The actual duration of rescheduling the 60 tasks was approximately 23 s, which is 3 s greater than estimated. As such, the decision to reschedule was not affected.

During rescheduling, the activity directors administer the enactment of their respective interim schedule models. Because the tasks included within the interim schedule models have no outstanding dependencies, the need for dependency checking is not required. The omission of dependency checking is essential because it requires scheduler involvement, which is not possible because this agent is occupied performing rescheduling during the enactment of the interim schedule models.

4.2.16. Complete revised schedule models

Once rescheduling has been performed and the interim schedule models are completed simultaneously, then the revised schedule models are enacted. The computational process then progresses until all tasks have been completed in accordance with the revised schedule models.

In the application of the DCS presented in this article, the time taken to complete one run of the computational process is approximately 3 min, while utilizing four workstations of variable performance efficiency. As stated at the outset of Section 4, an experienced engineer can complete one run of the process in approximately 8 min using a single workstation. However, it is not only the additional workstations employed by the DCS that contributes to the reduction in the duration of the computational process but also the approach to real-time operational design coordination. That is, the techniques and their links and interrelationships within the approach demonstrate the coherent management of tasks, resources, and schedules can result in the computational process being performed in a timely and appropriate manner. More specifically, the integrated approach provides a means of undertaking tasks in a structured fashion, and resource utilization is continuously optimized throughout the computational process, in accordance with appropriately and dynamically generated schedules, within a computer environment that is susceptible to fluctuations in performance at any time. Indeed, Siemens Power Generation Limited indicated in correspondence that, "while there are obvious benefits of automating the process involving the use of the analysis tools on a network of machines, I appreciate that it is the underlying real-time operational design coordination approach and the work in setting up the architecture for this that is of key importance" (Coates, 2001).

5. DISCUSSION

This section presents a discussion of the approach based on the application of the DCS to the industrial case study. As a result of the discussion, strengths and weaknesses of the approach are identified. With regard to the DCS, in Section 3.1, it was stated that "the composition of agents, along with the role each fulfills, enables them to communicate with each other in real time such that they can perform activities involving task management, resource management and schedule management simultaneously in a coherent manner." As such, in this section, the key elements of coherence, communication, and real-time support are not covered explicitly but, rather, are discussed in the context of managing tasks, resources, and schedules.

5.1. Task Management

In terms of task management, the approach aims to organize and control tasks and their dependencies such that they can be undertaken in a structured manner.

5.1.1. Construction and management of the task model

The construction of the task model, and the subsequent management of the knowledge held within it, provided the basis for the structured undertaking of tasks. Initially, tasks were established based on knowledge provided by the designer regarding the analysis tools within the computational process. Tasks and their associated analysis tools were assigned unique indices (i.e., T_I, T_L, and T_G), such that they could be identified within the application of the DCS. Knowledge of each task's datum duration (T_{DD}) was used such that, once scheduled, the estimated duration (T_{ED}) of the task could be determined on the basis of the forecasted performance efficiency of the assigned resource. Knowledge of the completion of a task (i.e., T_C) was required for purposes of rescheduling such that once a task was completed it could not be considered again. Establishing knowledge of dependencies involved the construction and use of an analysis tool dependency matrix. Using the matrix, dependency knowledge was established by comparing the input requirements of each task's associated analysis tool ($T_{[T_{In}]}$) with the output requirements of all other task's associated analysis tool ($T_{[T_{Out}]}$). During the computational process, knowledge of dependencies was checked when and where appropriate such that tasks could only be undertaken if their dependencies had been completed.

Preparation for rescheduling involved the task model being modified such that only the tasks to be included within the revised schedules would be considered. Further, knowledge attributes of these tasks, and their dependencies, were modified such that revised schedules would accurately reflect the outstanding tasks. In addition, although not yet undertaken, completion indicators were modified regarding the tasks to be completed during rescheduling (i.e., within the interim schedule models).

5.1.2. Information management

Within the DCS, the organization, provision, and storage of task input/output information was supported such that tasks could be undertaken when required. A consequence of managing task dependencies was the assurance that in the event of a task being required to be undertaken, any necessary information was available. When a task manager requested information to undertake a task, the necessary input files were retrieved from the task information repository and provided by the related information manager. On completion of a task, the resulting output files were stored by the relevant information manager so that they could be retrieved in the future if needed. This process of requesting, providing, and supplying task information was performed for every task within the computational process. As a result of managing information in this manner, in no case was information not provided on request because it was not available.

5.2. Resource management

In relation to resource management, the approach aims to organize and control resources to enable their continuous optimized utilization throughout the computational process. Resource management in the case study is assessed by considering resource utilization during the computational process in terms of the enactment of the original, interim and revised schedules.

5.2.1. Resource utilization during the enactment of the original schedule model

The DCS supported the forecasting of resource performance efficiency to aid the derivation of schedules such that resource utilization could be optimized. Before the original schedule was derived, each resource monitor forecasted performance efficiency of their associated resource using regression analysis and orthogonal polynomials. These forecasts were then supplied to the resource manager, which updated the resource model. As a consequence, the original schedule produced enabled the optimized utilization of the resources because the appropriate type and number of tasks were assigned to them.

Initially, resources were used in an optimized manner for a proportion of the original schedule models. However, at a certain point in time (i.e., $t = 60$ s), the monitored performance efficiency of resource $R_I = 4$ began to decrease, which change was observed by the associated resource monitor. Thus, although the monitored performance efficiencies of three of the four resources closely adhered to their original forecasted performance efficiencies, resource $R_I = 4$ started to affect their utilization. Because of this decrease, tasks expected to be undertaken on resource $R_I = 4$ were prolonged. In addition, dependencies of these tasks were delayed themselves, thus affecting the utilization of the other resources. Although resource $R_I = 4$ continued to be utilized, rescheduling was considered and, because it was appropriate, was performed to avoid any further delays being encountered. As such, resources were utilized in an optimized manner for the proportion of the original schedule models that were enacted. In addition, the timely recognition of the need to consider and perform rescheduling ensured that the future utilization of resources could also be optimized.

5.2.2. Resource utilization during the enactment of the interim schedule model

In considering whether or not to reschedule, interim schedule models were derived by applying an iterative procedure, rather than using the MOGA. The application of this procedure resulted in tasks being included within the interim schedule models that would ensure, as near as possible, that resources would be utilized in an optimized manner during rescheduling. The interim schedule models reflected the tasks able to be completed by each of the resources, taking into account their respective newly forecasted performance efficiencies. Because rescheduling was performed, the interim schedule models enabled the continuation of optimized resources utilization during this period. Furthermore, because of the appropriate division of outstanding tasks being rescheduled and being included within the interim schedule, the completion of and was near coincident, thus minimizing resource idle time between the original and revised schedules.

5.2.3. Resource utilization during the enactment of the revised schedule model

As with the original schedule, the revised schedule was derived using the MOGA. Thus, the revised schedule models reflected the tasks able to be completed by each of the resources considering their respective newly forecasted performance efficiencies. In the application of the DCS, the revised schedule models were completed in the time estimated because the forecasts of performance efficiency approximately corresponded to those subsequently monitored during the remainder of the computational process. As such, resources were utilized in an optimized manner during the enactment of the revised schedule models.

5.3. Schedule management

In Section 1, schedule management was defined as "managing the dynamic assignment of tasks to resources, and the enactment of the resulting schedules."

5.3.1. Dynamic scheduling

Within the DCS, dynamic scheduling involves accessing appropriate task and resource knowledge for use with the MOGA. This knowledge was maintained within the task and resource models throughout the operation of the DCS. Thus, when scheduling/rescheduling was performed, the

resulting schedules ensured that tasks were undertaken in a structured manner and resource utilization was optimized.

5.3.2. Schedule enactment/pending scheduled tasks

To facilitate the structured undertaking of tasks, the enactment of schedules was supported by managing tasks and the dependencies between them. During the computational process, three types of schedules were enacted (i.e., original, interim, and revised).

The procedure for enacting the original and revised schedule models involved the relevant agents ensuring that tasks could be undertaken by checking dependencies and providing the required information. However, in situations in which tasks could not commence when scheduled, because of dependencies not being completed, the pending schedule task repository provided support such that they were only undertaken when appropriate. Thus, on the completion of every task, in addition to updating the task model, the pending scheduled task repository was checked and, if appropriate, any awaiting tasks could commence. As a consequence, throughout the computational process, no situation occurred in which a task was attempted to be undertaken when it was not possible.

Enacting the interim schedule models differed from that of the original and revised schedule models in that task dependency checking was not required. The reason for omitting this action was that interim schedule models only included independent tasks. By omitting this checking during the enactment of the interim schedule models, the scheduler was free to perform rescheduling uninterrupted, and the task model was not altered such that it would effect the enactment of the revised schedule models.

5.3.3. Decision making for rescheduling

Before making the decision regarding whether or not to reschedule, the scheduler instructed each activity director to suspend the associated original schedule model. At the point in time when suspension was instructed, 3 tasks were being undertaken and 1 was pending. Because of tasks being nonpreemptive, the 3 tasks being undertaken were completed. However, the pending task was not undertaken, and knowledge of this task was removed from the pending scheduled task repository. This removal was required because failure to do so would cause deadlock on the enactment of the following interim or revised schedule models. Once all activity directors had suspended the enactment of their associated original scheduled models, 27 tasks had been completed and 104 remained outstanding.

To ensure that rescheduling was only undertaken if appropriate (i.e., would lead to a reduction in the time taken to complete the computational process) time estimations were determined for the completion of the original schedule, derivation of the revised schedule, and completion of the revised schedule. The scheduler made the decision to reschedule because a revised schedule could be derived and completed in an estimated 38 s, whereas continuing to adhere to the original schedule models would take an estimated 79 s. As such, these estimations provided in excess of a 50% reduction in time to complete the computational process from the point at which rescheduling was considered. This feature of the approach demonstrates that by adjusting in real time when appropriate in a coordinated manner, benefits can be made in terms of reducing the time to complete the computational process.

5.3.4. Concurrent rescheduling and undertaking tasks

An outcome of determining the estimated time to derive a revised schedule was the identification of those tasks that could be undertaken during the period of rescheduling (i.e., included within the interim schedule models). In addition to the identification of these tasks, the DCS supported their undertaking during the period of rescheduling. Further, application of the procedure used to determine the tasks for inclusion within the interim schedule models resulted in their completion occurring within several seconds before the arrival of the revised schedule.

5.4. Strengths and weaknesses

The integrated approach realized within the DCS was shown to coordinate a computational process in real time through agents communicating and interacting with each other coherently. That is, the agent composition within the DCS ensured the continuous optimized utilization of resources by adapting to unforeseen changes in resource performance efficiency. Further, communication between agents was meaningful, enabling them to interact in a nonchaotic fashion, leading to resource effort and tasks being integrated harmoniously. In terms of the management of tasks, resources, and schedules, the approach integrated the respective mechanisms within a unified manner to facilitate the computational process to be coordinated at an operational level.

The approach manages and models tasks, and the dependencies between them, such that any schedule derived will ensure that they can be undertaken in a structured manner, ceasing opportunities for concurrency when and where appropriate. The structured undertaking of tasks is also ensured through the organization, provision, and storage of information. A weakness of the approach is that it only caters for tasks that cannot be interrupted if they are to be successfully completed. The approach, and the DCS, would need development to be able to manage tasks such that they could be suspended at any point during their undertaking. Further, incorporating this development would affect the integration between task management and aspects of schedule management.

The approach enables resource utilization to be continuously optimized throughout the computational process despite the occurrence of changes in performance efficiency. This is achieved through monitoring resources and, thus, detecting significant discrepancies in performance effi-

ciency that violate defined thresholds, which are then reported in order to activate the consideration of rescheduling. Further, the forecasting of resource performance efficiency is enabled, as and when required, to assist in the derivation of more appropriate schedules such that the optimized utilization of resources can be maintained. However, the forecasting technique used only allows a prediction of resource forecasted performance efficiency to be made at one time step ahead.

Dynamic scheduling and the subsequent enactment of derived schedules is enabled within the approach such that resource utilization can be continuously optimized with respect to the structured undertaking of interrelated tasks. In addition, decision making ensures that rescheduling only occurs if appropriate (i.e., will lead to a reduction in time to complete the computational process). Finally, the approach provides a mechanism allowing concurrent rescheduling and undertaking of tasks such that their completion is near coincident. As such, this mechanism facilitates minimum transition delay between adjacent schedules and also ensures that resource inactivity is minimized. A shortcoming of this mechanism is that the interim schedule models derived only consist of tasks that are independent at the time of being considered for inclusion. The procedure of determining which outstanding tasks could be included within the interim schedule models would need to be developed for them to include dependent tasks.

6. SUMMARY AND FUTURE WORK

Efficient design management is recognized as a means of ensuring that engineering companies remain competitive. Within contemporary approaches to design management, coordination has been identified as an important and pervasive characteristic. However, there exists a broad and varied understanding of coordination. Rather than being regarded as a ubiquitous characteristic of other approaches, this article focuses on coordination as the foundation of an improved approach. The ethos of coordination is to do the right thing at the right time for the right reasons. Thus, the focus is not on cooperation or concurrency but on optimizing the design process with respect to timeliness and appropriateness. On the basis of literature dedicated to coordination within several disciplines, including engineering design, knowledge of the key elements of operational design coordination has been established: coherence, communication, task management, resource management, schedule management, and real-time support.

A novel integrated approach to operational design coordination is the main contribution of the work presented in this article. The approach is founded on a more comprehensive set of key elements of operational design coordination than previously identified. As such, the approach provides knowledge of the constituent techniques of operational design coordination, the interrelationships and dynamic interactions between the techniques, and the knowledge used and maintained within and between the techniques. It is not only do the individual techniques themselves that define the approach but also, more significantly, the interrelationships and interactions that enable them to be integrated. Further, as the individual problems, which are associated with the respective key elements of operational design coordination, are not mutually exclusive, the techniques employed within the approach have been developed simultaneously and in consideration of each other. Thus, techniques have been integrated such that the approach exercises real-time operational design coordination to achieve the coherent, timely, and appropriate structured undertaking of interrelated tasks while continuously optimizing the utilization of the resources in accordance with dynamically derived schedules. The approach is supported by the appropriate modeling of the knowledge attributes of tasks and resources.

An agent-oriented computer-based system, the DCS, has been developed to facilitate the application of the approach. The DCS architecture provides knowledge of how the approach is realized within an agent-oriented system. That is, knowledge is provided regarding how task, resource, and schedule management responsibilities are attributed among the agents within the DCS and, further, how the agents communicate and interact, enabling a computational process to be performed in a coherent manner in real time. As such, the novelty of the approach lies in the integrated style of simultaneously managing the complexities of the structured enactment of tasks, continuous optimized utilization of resources, and appropriate dynamic scheduling. That is, the approach both enables and demonstrates how resources can be utilized in an optimized fashion throughout a dynamic process and environment, dependencies between tasks are managed in real time such that tasks can be undertaken at the appropriate time while seizing opportunities for concurrency when and where possible, and schedules can be managed in real time and decisions made to determine whether to dynamically generate new schedules or to continue adhering to a current schedule. In addition, the approach as a whole consists of the appropriate mechanisms to ensure that these items are managed irrespective of when changes happen during the process or where they occur in the environment. Further, the novel configuration of agents within the DCS enables them to operate coherently in real time through meaningful communication and interaction. Jointly, these novel aspects provide a more efficient approach to coordination than those already in existence.

The integrated approach, using the DCS, has been applied to a computational process of turbine blade design. The application of the system has been shown to support the coherent, timely, and appropriate communication and interaction between the various DCS agents that have designated responsibilities with regard to the management of tasks, resources, and schedules. That is, throughout the computational process, agents adapt to changes such that the

right tasks are undertaken using the right information while utilizing the right resources at the right time.

Within the approach, tasks are modeled and managed such that they can be undertaken in a structured manner. This is assisted by managing the dependencies and information required to undertake such tasks. Through the appropriate modeling and management of resources, the approach enables and sustains their continuous optimized utilization throughout the computational process, even in the event of unforeseen changes in performance efficiency. This is achieved by monitoring, forecasting, and disseminating resource performance efficiency as and when necessary. On the basis of knowledge of the tasks to be undertaken and the resources to be used, the approach incorporates schedule management, which involves performing dynamic scheduling and enacting the resulting schedules. Further, the approach provides decision-making support for rescheduling such that schedules are only created if appropriate in terms of reducing the time to complete outstanding tasks. In addition, if rescheduling is performed, it is done so in parallel with undertaking tasks, thus making best use of the resources during this period.

Future work will be directed toward theoretical improvements to the approach, further developments of the DCS, and further applications of the approach. Theoretical improvements to the approach will involve enabling coordinated utilization of resources before the derivation of an original schedule and developing a method to manage the complexities involved in enabling interdependent tasks to be undertaken and completed during rescheduling. Further developments to the DCS are proposed as providing a means of forecasting over longer ranges and of reimplementation in a platform independent programming language to enable multiplatform operational design coordination. Finally, relatively significant reductions in the time to complete the computational process have been achieved as a result of applying operational design coordination in real time. To demonstrate the approach further in terms of scalability, it should be applied within an engineering company to establish whether similar savings in time could be achieved on a larger absolute scale (i.e., on the order of human weeks or months).

ACKNOWLEDGMENTS

We thank Siemens Power Generation Limited for supplying the case study.

REFERENCES

Alonso, G., Agrawal, D., & El Abbadi, A. (1996). Process synchronization in workflow management systems. *Proc. Eighth IEEE Symp. Parallel Distributed Processing*, pp. 581–588.

Andreasen, M.M., Duffy, A.H.B., MacCallum, K.J., Bowen, J., & Storm, T. (1996). The design co-ordination framework: key elements for effective product development. *Proc. First Int. Engineering Design Debate*, pp. 151–174.

Bailetti, A.J., Callahan, J.R., & DiPietro, P. (1994). A coordination structure approach to the management of projects. *IEEE Transactions of Engineering Management, 41(4)*, 394–403.

Bendeck, F., Goldmann, S., Holz, H., & Kotting, B. (1998). Coordinating management activities in distributed software development projects. *Proc. Seventh Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 33–38.

Bennett, F.L. (1996). *The Management of Engineering*. New York: Wiley.

Cleetus, K.J., Cascaval, G.C., & Matsuzaki, K. (1996). PACT—a software package to manage projects and coordinate people. *Proc. Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 162–169.

Coates, G. (2001). *An approach to operational design co-ordination*. PhD Thesis. Newcastle upon Tyne, UK: University of Newcastle upon Tyne.

Coates, G., Duffy, A.H.B., Hills, W., & Whitfield, R.I. (1999). Enabling concurrent engineering through design coordination. *Proc. Sixth Int. Conf. Concurrent Engineering: Research Applications*, pp. 189–198.

Coates G., Whitfield R.I., Duffy A.H.B., & Hills W. (2000). Coordination approaches & systems—part II: an operational perspective. *Research in Engineering Design, 12(2)*, 73–89.

Cole, G.A. (1994). *Strategic Management*. London: DP Publications.

Cross, N. (1994). *Engineering Design Methods: Strategies for Product Design*. New York: Wiley.

Davis, M.B., & Sydir, J.J. (1996). Resource management for complex distributed systems. *Proc. Second Workshop on Object-Oriented Real-Time Dependable Systems*, pp. 113–115.

Decker, K.S., & Lesser, V.R. (1995). *Coordination Assistance for Mixed Human and Computational Agent Systems*. Computer Science Tech. Report 95-31. Boston: University of Massachusetts.

de Jong, E. (1997). Multi-agent coordination by communication of evaluations. *Proc. Eighth Eur. Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent Rationality*, pp. 63–78.

Dellen, B., & Maurer, F. (1996). Integrating planning and execution in software development processes. *Proc. Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 170–176.

Du, W., & Shan, M.-C. (1999). Enterprise workflow resource management. *Proc. Ninth Int. Workshop on Research Issues in Data Engineering: Information Technology for Virtual Enterprises*, pp. 108–115.

Duffy, A.H.B. (1998). An approach to developing design technology. *IEE Colloquium—Design Technology: An Integrated Approach to Design of a T&D Plant*.

Duffy, A.H.B., Andreasen, M.M., Bowen, J., MacCallum, K.J., & Reijers, L.N. (1994). Design coordination support. *Esprit Basic Research Working Group 7401—CIMDEV/CIMMOD Workshop*, pp. 1–34.

Duffy, A.H.B., Andreasen, M.M., MacCallum, K.J., & Reijers, L.N. (1993). Design coordination for concurrent engineering. *Journal of Engineering Design, 4(4)*, 251–265.

Duffy, A.H.B., Andreasen, M.M., & O'Donnell, F.J. (1999). Design co-ordination. *Proc. Twelfth Int. Conf. Engineering Design, 1*, 113–118.

Durfee, E.H., & Montgomery, T.A. (1990). A hierarchical protocol for coordinating multiagent behaviors. *Proc. Eighth Natl. Conf. Artificial Intelligence*, pp. 86–93.

Durfee, E.H., & So, Y.P. (1997). The effects of runtime coordination strategies within static organisations. *Proc. Fifteenth Int. Joint Conf. Artificial Intelligence*, pp. 612–619.

Eppinger, S.D. (2001). Innovation at the speed of information. *Harvard Business Review, 79(1)*, 149–158.

Eppinger, S.D., Whitney, D.E., Smith, R.P., & Gebala, D.A. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design, 6*, 1–13.

Fayol, H. (1949). *General and Industrial Management*. London: Pitman.

Findler, N.V., & Elder, G.D. (1995). Multiagent coordination and cooperation in a distributed dynamic environment with limited resources. *Artificial Intelligence in Engineering, 9*, 229–238.

Finlay, P. (2000). *Strategic Management*. Englewood Cliffs, NJ: Prentice Hall.

Goldmann, S. (1996). Procura: a project management model of concurrent planning and design. *Proc. Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*.

Greenley, G.E. (1989). *Strategic Management*. Englewood Cliffs, NJ: Prentice Hall.

Hansen, P.H.K. (1995). *Computer Integration: A Co-Requirement for Efficient Organizational Coordination*. Amsterdam: IOS Press.

Harrison, F.L. (1992). *Advanced Project Management: A Structured Approach*. New York: Gower.
Hayden, S.C., Carrick, C., & Yang, Q. (1999). A catalog of agent coordination patterns. *Proc. Int. Conf. Autonomous Agents*, pp. 412–413.
Jamali, N., Thati, P., & Agha, G.U. (1999). An actor-based architecture for customizing and controlling agent ensembles. *IEEE Intelligent Systems, 14(2)*, 38–44.
Jennings, N.R. (1996). Coordination techniques for distributed artificial intelligence. In *Foundations of Distributed Artificial Intelligence* (O'Hare, G.M.P., & Jennings, N.R., Eds.), pp. 187–210. New York: Wiley.
Jennings, N.R., & Jackson, A.J. (1995). Agent-based meeting scheduling: a design and implementation. *IEE Electronics Letters, 31(5)*, 350–352.
Jin, Y., & Levitt, R.E. (1996). The virtual design team: a computational model of project organizations. *Journal of Computational and Mathematical Organization Theory*, *2(3)*, 171–195.
Kleinman, D.L. (1990). Coordination in human teams: theories, data and models. *Proc. Eleventh Triennial World Congr. Int. Federation of Automatic Control*, pp. 417–422.
Kunz, J.C., Christiansen, T.R., Cohen, G.P., Jin, Y., & Levitt, R.E. (1998). The Virtual Design Team: a computational simulation model of project organizations. *Communications of the Association for Computing Machinery*, *41(11)*, 84–91.
Kusiak, A., & Park, K. (1990). Concurrent engineering: decomposition and scheduling of design activities. *International Journal of Production Research, 28(10)*, 1883–1900.
Kusiak, A., & Wang, J. (1991). Concurrent engineering: simplification of the design process. *Computer Applications in Production and Engineering: Integration Aspects* (Doumeingts, G., Browne, J., & Tomljanovich, M., Eds.), pp. 297–304. New York: Elsevier.
Kusiak, A., & Wang, J. (1993). Decomposition of the design process. *Journal of Mechanical Design, Transactions of the ASME, 115(4)*, 687–695.
Kwon, P., Chung, M.J., & Pentland, B. (2002). A grammar-based framework for integrating design and manufacturing. *ASME Journal of Manufacturing Science and Engineering, 124(4)*, 899–907.
Lesser, V. (1999). Cooperative multiagent systems: a personal view of the state of the art. *IEEE Transactions on Knowledge and Data Engineering, 11(1)*, 133–142.
Lesser, V.R. (1998). Reflections on the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems, 1*, 89–111.
Lock, D. (1993). *Handbook of Engineering Management*. London: Butterworth–Heinemann.
Lock, D. (1996). *Project Management*. New York: Gower.
MacCallum, K.J., & Carter, I.M. (1991). An opportunistic software architecture for support of design coordination. *Knowledge Based Systems Journal, 5(1)*, 55–65.
Malone, T.W. (1987). Modeling coordination in organisations and markets. *Management Science, 33(10)*, 1317–1332.
Malone, T.W., & Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Surveys, 26(1)*, 87–119.
McCord, F.P., & Eppinger, S.D. (1993). *Managing the Integration Problem in Concurrent Engineering*. Working Paper 3594-93-MSA. Cambridge, MA: MIT.
Oberlender, G.D. (1993). *Project Management for Engineering and Construction*. New York: McGraw–Hill.
Pareto, V. (1896). *Cours d'Economie Politique*, Vol. 1. Lausanne, Switzerland: F. Rouge.
Pena–Mora, F., Hussein, K., Vadhavkar, S., & Benjamin, K. (2000). CAIRO: a concurrent engineering meeting environment for virtual design teams. *Artificial Intelligence in Engineering, 14*, 203–219.
Perrin, J. (1997). Institutional and organizational prerequisites to develop cooperation in the activities of design. *Proc. Eleventh Int. Conf. Engineering Design*, Vol. 1, pp. 87–92.
Piccinelli, G. (1998). *Distributed Workflow Management: The TEAM Model*. Tech. Report No. 98-56, pp. 1–17. Bristol, UK: Hewlett–Packard Laboratories.
Pimmler, T.U., & Eppinger, S.D. (1994). Integration analysis of product decompositions. *Proc. Sixth Int. Conf. Design Theory Methodology*, pp. 343–352.
Pourbabai, B., & Pecht, M. (1994). Management of design activities in a concurrent engineering environment. *Production Research, 32(2)*, 821–832.
Prasad, B. (1996). *Concurrent Engineering Fundamentals, Vol. 1: Integrated Product and Process Organization*. Englewood Cliffs, NJ: Prentice Hall.
Ray, M.S. (1985). *Elements of Engineering Design: An Integrated Approach*. Englewood Cliffs, NJ: Prentice Hall.
Schal, T. (1996). *Workflow Management Systems for Process Organizations*. New York: Springer.
Tan, G.W., Hayes, C.C., & Shaw, M. (1996). An intelligent-agent framework for concurrent product design and planning. *IEEE Transactions of Engineering Management, 43(3)*, 297–306.
Thamhain, H.J. (1992). *Engineering Management*. New York: Wiley.
Todd, D. (1997). *Multiple Criteria Genetic Algorithms in Engineering Design and Operation*. PhD Thesis. Newcastle upon Tyne, UK: University of Newcastle upon Tyne.
Todd, D., & Sen, P. (1997*a*). A multiple criteria genetic algorithm for containership loading. *Proc. Seventh Int. Conf. Genetic Algorithms*, pp. 674–681.
Todd, D., & Sen, P. (1997*b*). Multiple criteria scheduling using genetic algorithms in a shipyard environment. *Proc. Ninth Int. Conf. Computer Applications in Shipbuilding*, pp. 234–265.
Wallace, K.M. (1987). Studying the engineering design process in practice. *Int. Congr. Planning Design Theory: Plenary Interdisciplinary Lectures*, pp. 29–34.
Whitfield, R.I., Coates, G., Duffy, A.H.B., & Hills, W. (2000*a*). Co-ordination approaches & systems—part I: a strategic perspective. Research in Engineering Design, 12(1), 48–60.
Whitfield, R.I., Coates, G., Duffy, A.H.B., & Hills, W. (2000*b*). A multiagent based system to enable strategic and operational design co-ordination. *Workshop on Developing Intelligent Support for Collaboration in Design, Sixth Int. Conf. Artificial Intelligence in Design*.
Whitfield, R.I., Duffy, A.H.B., Coates, G., & Hills, W. (2002). Distributed design co-ordination. *Research in Engineering Design, 13(4)*, 243–252.
Wilson, J.L., & Shi, C. (1996). Coordination mechanisms for cooperative design. *Engineering Applications of Artificial Intelligence, 9(4)*, 453–461.
Yu, L. (1996). A coordination-based approach for modelling office workflow. *Proc. Fifteenth Int. Conf. Conceptual Modelling*.

Graham Coates is a Lecturer in the School of Engineering at the University of Durham. Prior to this, he was a Senior Research Associate in the Engineering Design Centre at the University of Newcastle upon Tyne, where he conducted research in the area of operational design coordination and robust/real-time production scheduling. Before this work he was employed as a Structural Engineer and Aerodynamicist in the aerospace industry for 7 years. During this time he worked on various certification aspects of turboprop aircraft and laminar flow technology for future aircraft.

Alex Duffy is a Reader in the Department of Design, Manufacture and Engineering Management and the Director of the CAD Centre at the University of Strathclyde. He lectures in engineering design, design management, product development, knowledge intensive CAD, and advanced computational techniques. His main research interests include the application of knowledge-based techniques, product and domain knowledge modeling, machine learning techniques, design reuse, performance measurement and design productivity, sketching of vague geometric modeling, and design coordination.

Ian Whitfield is a Research Fellow in the CAD Centre at the University of Strathclyde. Between 1987 and 1995 he attained a BEng(Hon) in mechanical engineering and a PhD in researching the modes of operation of turbogenerator rotor bearings. He then worked in the area of hydrodynamics and rotor dynamics at Parsons Power Generation Systems. Since 1996 Dr. Whitfield has worked on research projects related to robust design and strategic design coordination.

William Hills was the Director of the Engineering Design Centre and the Regional Centre for Innovation in Engineering Design at the University of Newcastle upon Tyne until his retirement in 1998. He has worked as a Principal Lecturer in Ship Design at the University of Sunderland and the University of Newcastle upon Tyne. During this period his main teaching and research interests included layout design and optimization of hull forms, particularly in the context of sea-keeping and small craft.

APPENDIX A

Table A.1. *Modeled task knowledge*

Knowledge Attributes	Description
T_{I}	Analysis tool identification index
T_{L}	Task identification index in the context of tasks associated with a specific analysis tool
T_{DD}	Datum duration of a task
$T_{[T_{\mathrm{In}}]}$	Matrix defining the input files for a task
$T_{[T_{\mathrm{Out}}]}$	Matrix defining the output files for a task
T_{G}	Task identification index in the context of all tasks
T_{C}	Completion indicator of a task
T_{N}	Number of dependencies of a task
$T_{[T_{\mathrm{G}}]}$	Matrix defining T_{G} of each dependency of a task
n_{TS}	Number of tasks to be scheduled
n_{TD}	Cumulative number of dependencies of the tasks to be scheduled
$T_{[T_{\mathrm{I}}]}$	Matrix defining T_{I} of each dependency of a task
$T_{[T_{\mathrm{L}}]}$	Matrix defining T_{L} of each dependency of a task
T_{ON}	Number of outstanding dependencies of a pending scheduled task
$[T_{\mathrm{I},i,j}]$	Matrix of the ith pending task defining T_{I} of its j dependencies
$[T_{\mathrm{L},i,j}]$	Matrix of the ith pending task defining T_{L} of its j dependencies
n_{PST}	Number of pending scheduled tasks
$T_{\mathrm{ON},i}$	Number of outstanding dependencies of the ith pending scheduled task
T_{ED}	Estimated duration of a task
n_{TRS}	Number of outstanding tasks to reschedule
n_{TCRS}	Number of tasks that could be completed using a particular resource during rescheduling
T_{TCRS}	Duration to complete a number of tasks using a particular resource during rescheduling

Table A.2. *Modeled resource knowledge*

Knowledge Attributes	Description
R_{I}	Identification index
R_{A}	Availability status
R_{FE}	Forecasted performance efficiency
R_{LT}	Lower performance efficiency threshold
R_{UT}	Upper performance efficiency threshold
R_{ME}	Monitored performance efficiency
R_{user}	CPU utilization attributed to user processes (i.e., computer programs being run by users) expressed as a percentage
R_{system}	CPU utilization attributed to system processes (i.e., UNIX kernel code) expressed as a percentage
R_{idle}	CPU utilization attributed to idle (i.e., not being utilized) expressed as a percentage
R_{DCS}	CPU utilization attributed to DCS processes (i.e., analysis tools being run within the DCS) expressed as a percentage
R_{other}	CPU utilization attributed to other processes (i.e., computer programs being run that are unrelated to the DCS) expressed as a percentage
R_{CF}	Coefficient (i.e., a relative measure of resource speed)
n_{ps}	Number of processes being run on a resource
n_{R}	Number of resources to be utilized in a schedule