Integration of Collaborative Approaches for Distributed Product Development

R.T. Sreeram and P.K.Chawdhry
Department of Mechanical Engineering
University of Bath, Bath
United Kingdom
Telephone: +44-1225-826826
Facsimile: +44-1225-829628
Email: {enprts, enspkc@bath.ac.uk}

Abstract

Research in distributed product development has identified two approaches: software agents and computer supported co-operative work (CSCW). Many of the existing product development environments treat these approaches independently that limit their use when applied to heterogeneous environments. This paper analyses the merits and limitations of the approaches individually, and evaluates the advantages offered by merging them in a common environment for product development. Our study is based on a collaborative product development architecture which consists of software agents, CSCW tools, human agents, and stand-alone manufacturing facilities. Our results are based on a case study of design and manufacture of a stepped shaft using plastic materials. Our main conclusion is that the integration of the software agent-based and CSCW approaches leads to an environment which has positive aspects of both approaches [eg., interoperability of legacy code, face-to-face meetings between humans]. The integration of collaborative approaches also leads to significant reuse of existing facilities and, reduction in development time.

1 INTRODUCTION

Recent years have witnessed a novel trend in global product development due to formation of Internet-based collaborative ventures. In distributed product development, partners with complementary competencies come together to design, manufacture and market a product. The potential of such business ventures is being realised as they reduce the time to-market a product. Research in distributed product development has identified two independent approaches based on software agents and CSCW. The software agent approach focuses on the synthesis of intelligent behaviour in machines to solve a problem. These software agents are highly goal-oriented and autonomous in nature. Their autonomous behaviour enables them to act independently. There are other agent characteristics which are explained later in the paper. Agents can be seen in a variety of applications such as intelligent web browser, media assistants, just to name a few [Ndumu and Nwana (1997)]. In contrast to the agent based approach for distributed product development, the CSCW technique relies on face-to-face communication between human agents. The attributes of face-to-face communication such as eye contact, gestures, and voice exchange increases mutual trust and understanding between partners [Cutkosky et al (1996)]. This approach is widely used in human dominated environments (e.g., virtual classrooms) due to its flexibility and ease of realising informal communication.

Previous research has addressed these approaches independently, particularly with respect to a product development environment [Erkes et al (1996), Hardwick et al (1996), Cutkosky et al (1996)]. Moreover, in these cases, prototype systems were developed for an ideal scenario where all facilities within the environment were assumed to be networkable. The present paper addresses collaborative product development from an integrated perspective thereby exploiting the merits of the two approaches. The product development environment considered in this study consists of software agents, CSCW tools, human agents and also stand-alone manufacturing facilities. An experiment on the design and manufacture of a stepped shaft was used as a vehicle to evaluate the benefits of the integration mechanism based on an extended Open System Interconnection (OSI) model developed by Sreeram and Chawdhry (1998).

The rest of the paper is organised as follows. Section 2 presents the requirement of a generic product development environment from technological and management perspective. Section 3 discusses the two collaborative approaches to distributed product development: software agents and CSCW and examines the extent to which the above technical and management requirements are met by these approaches. The interface between software agents and CSCW is presented in Section 4 considering three different product development scenarios: (a) software agent-based with no human involvement, (b) CSCW-based with human involvement and (c) Software agents, CSCW-tools, humans and stand-alone facilities. These scenarios represent respectively, a formal autonomous environment, an informal human dominated environment, and a typical hybrid environment for product development. A distributed product development architecture based on the hybrid collaborative approach is presented in Section 5 along with an experiment in distributed product development carried out on a research prototype. Section 6 presents the results of this study, followed by conclusions of this research.

2 REQUIREMENTS OF A PRODUCT DEVELOPMENT PROCESS

A product development process comprises a number of tasks which are formally arranged in an activity diagram using techniques such as IDEF0 [Chawdhry (1995)]. These activities are described using data, functions, methods and constraints. The methods can be

implemented in various ways, eg., an automated software program, an expert human decision process, a set of manufacturing operations, or a combination of these. In this section, we use the term *task agent* to represent any of these modes of implementing a method. A typical product development process is shown in Figure 1. The general requirements for such a distributed product development process can be classified into two major tracks: technological and managerial. The technological requirements address the specific infrastructural issues whereas the management requirements address the process logistic issues. The major technological requirements include:

Information sharing

In a collaborative process, information is shared between various heterogeneous task agents. The environment should provide robust interfaces to facilitate shared interaction. There is also a need for an information base to store information flow patterns during various stages of product development. Data hiding and modularity are considered as the required characteristics of an information sharing system. Data hiding is essential as it offers security, and hides the complexity. Modularity helps in efficient organising of design information that can be shared among various task agents during redesign.

Communication

Communication between task agents in a product development environment should be stable and precise. This is especially true in a heterogeneous environment where there is a need to use common ontologies which enable the shared understanding. The environment must support communication of product information in various formats and appropriate language must be specified for communication between task agents. In case of legacy systems, special communication protocols are needed to ensure proper processing of product information. Hardwick et al (1996) suggests the use of wrappers that provide data and functional interoperability between legacy/non-legacy systems.

Knowledge

Knowledge provides task agents the ability to infer and act in a given decision making situation. The term knowledge in product development environment are identified as design knowledge, process knowledge, conflict resolution knowledge and so on. A comprehensive knowledge library consisting of the necessary heuristics, algorithms and rules could be used to provide support at various stages of product development. This is necessary since knowledge is a representation of higher order concepts of data. In addition the represented knowledge should also provide transparency to the task agents in the environment.

Archiving

Capturing of information and knowledge during a product development process is necessary for back-tracking during redesign, reuse of previous partial solution and conflict resolution. An archiving mechanism should be capable of capturing the key decisions, dependencies between product data and support visualisation of product models when necessary. An efficient structuring of archived information is important for quick and precise retrieval of the desired information. The process of archiving is also referred to as creation of design history.

Other DFX support

The environment must also provide the necessary facilities for specific DFX analysis, manufacturing and testing the product for reliability and quality. Tools are also required to offer support to other activities such as engineering analysis, creation of design drawings, optimisation and simulation. The managerial requirements include:

Figure 1 A subset of a typical product development process

Task planning

Execution of independent tasks assists the automation and increases the overall autonomy of the environment. Each individual task can have many dependent sub-tasks and may require co-operation between several task agents. Thus role of each task agent must be precisely defined at the beginning of a process. The use of entity-relationship modelling techniques could provide a means to map tasks on to task agents.

Team formation

Team work is key to success in a product development environment. Efficient interaction among task agents is essential since many critical decisions result from group discussions. In some cases the problems are even geographically separated and require stable communication mechanisms to support team effort. In general for teams to be successful, the task agents in the environment must fully co-operate.

Conflict negotiation

Conflicts cannot be avoided during a product development process especially if this involves multiple perspectives. For a given conflict situation, choosing an appropriate negotiation strategy from a library of strategies is advantageous since this minimises the communication between conflicting task agents in most cases. Since the environment usually involves heterogeneous task agents, the conflict need not be always between entities of the same type. Mapping between all combination of conflict types and their negotiation strategies and storing this information is useful in approaching new conflicts.

Decision making

During task agent interaction in a product development process, each task agent is responsible for some decisions. These decisions must satisfy the functional and goal requirements. To facilitate the decision making process, a product development environment should provide tools to maintain a stable communication system, access to knowledge-base and co-ordination of activities.

Views

Each task agent acts based on its own intelligence and the experience from interacting with other task agents in the environment. Therefore, during a product development process, task agents may hold different view points for analysing a particular situation. The process models should be made flexible in order to accept and process the views put forward by different task agents.

The other managerial requirements include taking control over projects, safeguarding of intellectual property rights, flexibility and responsiveness. More information on managerial aspects of distributed product development can be found in Sreeram and Chawdhry (1997).

3 COLLABORATION APPROACHES

3.1 Software agents

The agent-based approach has emerged from the AI research which synthesises intelligent behaviour in machines. An agent is the one that can act in a dynamic problem solving environment. Software agents interact with other software or human agents, based on their own functionality and requirements of the situation. Agents communicate with other agents to share information and answer queries that may arise at various stages. In this paper, we have kept a focus on the agents relevant to the product development process.

Agent types include information agents (eg., Softbot, Digital libraries [Ndumu and Nwana, 1997]), task agents (eg., software agents for design, process planning), mobile agents (eg., Aglet work bench from IBM Corporation) and interface agents. The application of agents can be seen in a wide domain including telecommunications, electronic commerce, business process re-engineering and smart databases [Ndumu and Nwana (1997)]. Autonomy, Goal-oriented behaviour, communication ability, interoperability and pro-activeness are the main agent characteristics. Some of the software agent-based environments relevant to product development are presented next.

Frost and Cutkosky (1996) proposed an agent-based framework for product design and manufacturing over the Internet. The agents exchange product information through an agent communication language (similar to KQML) with well defined syntax and semantics. The agents presented by Frost and Cutkosky are fully autonomous agents. Erkes et al (1996) demonstrated the use of Internet and protocols such as CORBA to simulate manufacturing process and part fabrication. The manufacturing processes or part data are encapsulated as electronic design notebooks (EDNs) to exchange information through Internet.

The rich agent characteristics described above, coupled with the potential of network-centric computing such as Java could make way for future intelligent CAD/CAM agents that are more process-oriented than task-oriented. In this context, Deitz (1997) explains that efforts are underway to create intelligent software agents that could act as advisors and could provide meta level assistance to designers.

3.2 CSCW approach

The objective of CSCW is to enhance collaboration between humans on a network. The focus of CSCW is on virtual co-location of humans who are geographically separated. CSCW permits face-to-face communication between humans with exchange of gestures, voice and eye contact. Key elements of CSCW are multi-media communication, co-ordination of group activities, and real-time sharing of applications. The nature of interaction of interaction is typically synchronous with possible additions of asynchronous tools such as email.

Sharing product information over wide area networks is of concern especially if the data to be shared is incompatible between networked CAD systems. In such cases the information is shared using low-bandwidth asynchronous mode (eg., email, ftp, fax) as well as synchronous mode (eg., electronic notepad, video with whiteboard). The synchronous mode requires higher bandwidth when compared to the asynchronous mode. As the information is shared between human agents, consistency maintenance of shared information is an important issue and involves ordering of group operations in a session. For instance, Cutkosky, et al, (1996) presents an instance of the data transfer process between a CAD system (located in Michigan, USA) and a CAM system (located in Utah, USA). In this case, two points are worth noting: (a) The requirement was to generate a CNC milling program from a solid model in IGES format generated by the CAD system (b) Their results indicated that the data transfer process was much faster when faxing the model to the destination and build the CNC code from scratch than the transfer of IGES file from a remote location. It can be inferred that though in this case data interoperability was possible, it was efficient to choose the latter option due to time limitations. However, the ability of leading-edge CAD/CAM software such as STEP/EXPRESS, and EUCLID QUANTUM to leverage their unique capabilities of interoperating with legacy product models could significantly enhance the CSCW-based product development process [Deitz, (1997)].

The application of CSCW can be seen in both small and large collaborative environments such as virtual classrooms, digital libraries and virtual enterprises [Cutkosky et al (1996)]. Other CSCW issues include document management in a shared editing environment, version control, archiving and problems of poor quality audio and video transmission due to bandwidth limitations, all of which are relevant to distributed product development.

4 INTERFACE OF SOFTWARE AGENTS AND CSCW APPROACHES

Before addressing the interface of the two collaborative approaches, we first explore the merits and limitations of both software agents and CSCW approaches with respect to the technological and management requirements discussed in Section 2.

4.1 Merits and limitations of software agent approach

Agent-based approaches provide efficient and stable support to collaborative problem solving by hiding the complexity of specific problem solving from humans and at the same time delivering the results in a time-efficient manner [Ndumu and Nwana (1997)]. Their prominence is felt in most of the CAD/CAM software today due to their ability to interoperate with legacy systems based on known functions. Software agents provide efficient mechanisms for archiving of data and functional dependencies in a product development process thereby providing robust information base for reuse and redesign

[Sreeram and Chawdhry (1998)]. Having presented the major advantages of agent-based approach, we will now look at the features that the software agent technology fail to offer at this point of time.

Generally the software agents that are discussed in the literature tend to be task-oriented. In other words, they tend to be specific to a narrow domain. For example, the COSINE wine glass designer system discussed by Berker and Brown (1996) is an example of such systems. Research in other agent characteristics such as belief, emotions and desire are very much in the beginning stages and their appropriateness in product development has not yet been validated. Also, learning in software agents has not yet been studied completely [Ndumu and Nwana (1997)].

The role of humans in software agent-based product development is seen as passive due to agent autonomy, intelligence and the belief that the product development environment could survive without human involvement [Norman (1994)]. This causes additional concerns as the agents may not take into consideration the human acceptance of their autonomous actions. The degree of human control on a completely autonomous agent is thus very limited which can lead to a lack of confidence in its action. In addition to this, in a business process, communication with humans is essential for success. Intelligent software agent approach has yet to find satisfactory solutions to emulate human behaviour and informal interactions. It is argued that whereas a formal mechanism gives the agents precision and unambiguity, it also leads to lack of flexibility.

There may also be facilities in a product development environment which cannot be agentified due to a lack of networkable features. There are many legacy manufacturing facilities which are not suited for computer-based communication but still must be integrated into the environment due to resource limitations. In such circumstances the use of software agents alone in the environment does not provide a solution to integration of non-networkable facilities [Sreeram and Chawdhry (1998)].

4.2 Merits and limitations of CSCW approach

The primary benefit of using CSCW approach is that it facilitates face-to-face communication in a geographically distributed team. Face-to-face communication between humans increases trust and confidence and thereby strengthens the business relationship. Since CSCW approach primarily makes use of off-the-shelf tools, the time period required to build a CSCW-based environment is shorter compared to that for software agent-based environments. This is stated with the assumption that for most of the domains the software agents will have to be built from scratch as they tend to be specialised may not be readily available for the given domain. For short-term businesses this could result into a substantial financial burden.

However, the CSCW approach suffers from some major drawbacks due to informal means of information exchange, poor archiving and the tool performance largely dependent on the network bandwidth. Other drawbacks include the use of informal mechanisms for conflict negotiation and platform dependency of some commercially available CSCW tools [eg. InPerson - a Silicon Graphics based video conferencing tool].

4.3 Interface of collaborative approaches

Sections 4.1 and 4.2 highlighted the merits and limitations of the two collaborative approaches. It emerged that they seem to possess complementary strengths to overcome each other's limitations. Previous research in distributed product development has treated these approaches independently and has not addressed their integration. The integration of the two approaches is therefore necessary in most CAE environments where

(i) the full agentification of legacy manufacturing facilities is not possible, and (ii) humans play an active role in the decision making process.

In order to explain this rationale, we consider three simple product development environments as shown in Figure 3. Figure 3(a) represents a fully autonomous agent-based product development environment [PDE1]. Figure 3(b) shows an environment with human agents collaborating via CSCW tools [PDE2] and Figure 3(c) shows an environment where decisions are taken jointly by human agents and software agents [PDE3]. The common features in the three environments [PDE1, PDE2, PDE3] are domain ontology and a process model for product development.

Table 1 shows a comparison of the functionalities of the three environments. PDE1 provides robust mechanisms for communication based on a common ontology, and to generate time-efficient solutions for specific tasks. Also the interoperability between various agents at data and functional level is consistent as the agents communicate using a formal ACL syntax and semantics. PDE1 being a purely software agent-based environment, restricts human intervention within its process domain. This means the collaborating agents take decisions autonomously and hence could achieve only some of the functionalities as listed in Table 1. For example, in PDE1 the face-to-face communication between human agents is not possible as the software agents negotiate on behalf of humans. PDE1 presents efficient mechanisms for archival and conflict negotiation.

PDE2 offers good support for the process but communication between human agents is facilitated by video conferencing tools that depend on the network bandwidth. Hence the communication mechanism is not as robust and formal as in PDE1. But the PDE2 approach certainly facilitate face-to-face communication which was absent in PDE1. Therefore PDE2 offers more benefits from a managerial perspective. In the case of PDE2 the main drawbacks are due to lack of proper archiving of proceedings of a session and conflicts between human agents are non-deterministic.

In the case of PDE3, the environment consists of humans, software agents, CSCW tools and stand-alone manufacturing facilities. This is the typical class of environments which offers both human participation and software agent capabilities.

Figure 2. Product development environments: (a) An autonomous decision process, (b) A human team-based decision process, and (c) A combined human and software-based decision process

Table 1 Comparison of attributes in PDE 1, PDE2 and PDE3

	PDE1	PDE2	PDE3
Autonomy	Full	Partial	Partial
Adaptability	Full - Partial	Full - Partial	Full - Partial
Archiving	Formal	Informal	Formal + Informal
Communicative action	Formal	Informal	Formal + Informal
Communication efficiency	Predictable	Bandwidth-limited	Limited and Predictable
Conflict handling	Formal	Informal	Formal + Informal
Face-to-face communication		Full	Partial
Feel of control		Full	Full-Partial
Goal-oriented behaviour	High	High-Low	High-Low
Information sharing	Precise	Fuzzy	Precise + Fuzzy
Interoperability	Full - Partial	Partial	Full-Partial
Learning ability	Average	Good	Average-Good
Ontology	Formal	Informal	Formal + Informal
Process visualisation		Full	Full-Partial
Pro-activeness	High	High	High
Trust		High-Low	High-Low

Thus one can realise the complementary benefits of merging the two collaborative approaches. Moreover, the presence of humans can provide interoperability between all networked and non-networked agents in the environment as shown in Figure 3(c) [Sreeram and Chawdhry (1998)]. Therefore it can be summarised from Table 1 that PDE3 gains the positive features of software agents and CSCW tools. It also addresses the issue of inclusion of humans in product development as seen in most of the present day environments. A product development architecture to realise a PDE3-type environment is presented next.

Figure 3. Product development architecture and the extended OSI model

5 A PRODUCT DEVELOPMENT MODEL AND A COLLABORATION ARCHITECTURE

The product development process is based on the extended OSI model developed by Sreeram and Chawdhry (1998). The extended OSI model is shown in Figure 4(a). The layers (1-4) of the OSI model deal with the installation of network and other protocols such as TCP/IP. Layer (5) deals with the protocols for data exchange such as CORBA and Hypertext Transfer Protocol (HTTP). Therefore layer (5) provides interoperability between heterogeneous software agents in the network. Layer (6) is the presentation layer which deal with the wrappers for applications. Wrappers enable legacy applications to become collaboration aware and facilitate data and functional interoperability between them. Layer (7) is the application layer which consists of both networked and non-networked objects.

In order to realise the OSI model, a product development architecture is proposed which is configured as shown in Figure 4(b). As shown in Figure 4(b), the product development environment consist of networked and non-networked facilities. The networked part automatically brings the attributes of layers (1-4) of the OSI model. The integration of the non-networkable facility is provided by including a human wrapper in layer (6) of the OSI model. More information on the architecture can be found in Sreeram and Chawdhry (1998).

6 EXPERIMENTAL EVALUATION OF THE COLLABORATION ARCHITECTURE

6.1 Objective

One of the main objectives of this experiment was to analyse the effectiveness of the above architecture as a mechanism for integration of agent-based and CSCW approaches.

6.2 Hypothesis

The software agent-based and CSCW approaches collectively offer sufficiently rich solutions that enable us to realise the attributes of product development environments as mentioned in Section 4.

6.3 Experiment

The experiment consisted of three main phases: team formation, assembling the test bed and product development. The design and manufacture of stepped shaft was chosen as the delivery vehicle for this experiment.

6.3.1 Team formation

This phase involved face-to-face meetings between team members for the first time in order to develop working acquaintance. This was done to establish trust and understanding between partners. During this phase the facility requirements, availability of resources and various tasks were identified by the team members.

6.3.2 Test bed assembly

Before the actual experiment was conducted, the various tools shown in Figure 4(b), Section 5 were tested for their performance. This also helped the human participants to become familiar with the tools.

6.3.3 Networked product development

This phase covers the complete design and manufacturing of the stepped shaft. The experimental set-up is shown in Figure 4(a). The human agent HA1 was responsible for specifying the load specifications for the shaft to the load agent. The load and material information was sent to the design and process planning agent (DPPA) as KQML messages. The evaluated the safe design for the shaft based on the constraints specified by HA1. The proceedings were shared with other human agents using a video conferencing tool, InPerson. The conflict negotiation mechanism within the software agents was based on constraint satisfaction and relaxation [Berker and Brown (1996)]. The conflicts between human agents were resolved informally by the use of InPerson and through email. The human agent HA2 acted as the wrapper for the stand-alone injection moulding machine. HA2 was also responsible for mould design. The material agent was responsible for supplying the necessary material data to HA2. The mould fabrication was accomplished by HA3 and HA4 and the result is shown in Figure 5(a). The product was then fabricated on the injection molding machine using a grade of polypropylene and the result is shown in [Figure 5(b)]. The human agents HA1, HA2 and HA5 then performed the post-processing of the manufactured product for any defects, dimensional accuracy, safety and quality. Each stage of the process was archived using WWWeasel, a tool developed by Enterprise Integration Technologies, Inc., USA, for distributed document generation.

(a) One half of the mould

(b) Fabricated Product

Figure 4. Mould and the product

6.4 Evaluation of integration

During this experiment, a variety of information was exchanged between the agents. Based on our observations the attributes that were gained by merging are presented in Figure 6. The grey areas indicate the particular attributes that were not realised during the experiment.

During the team forming stage the CSCW tools provided the face-to-face interaction in the environment. Thus we could establish trust, feel of control and confidence among human agents. During the problem specification stage, the interaction was mainly between human and software agents. At this stage, the effectiveness of communication (both formal and informal) was tested. The communication between a human agent and a software agent was formal by the use of well defined ontologies. The informal communication occurred between human agents via the use of InPerson and email for discussing product specification such as the load characteristics of the shaft. The human agents found the informal communication useful as this increased the positive interaction between team members while maintaining their autonomy and feel of control over specified activities.

The autonomy of software agents was realised during the product design, and preliminary process planning stages where there was the least human intervention in the activities of the software agents. The effectiveness of formal communication between software agents was observed as they exchanged messages using KQML with well defined ontologies. However, the vocabulary of the software agents for communication was limited to a pre-defined set of KQML performatives. Communication between software agents also enabled the sharing of engineering data between them. The software agents present in the environment can be considered as fully autonomous with respect to their domain. For example, the decisions of DPPA during the product design stage was not affected by humans and DPPA had the autonomy to reject request from other agents using KQML performatives such as *disagree* and *tell* if the requested item was not appropriate.

The level of interoperability between software agents themselves was fully possible as they exchanged semantically compatible messages. The level of interoperability between the software agents and the manufacturing machine (injection molding machine) during the process planning stage could not proceed without human intervention due to the non-networkability of the machine in the environment. Therefore a human wrapper was provided to the injection molding machine in order to facilitate data and functional interoperability between software and the machine.

The human agents were able to visualise the process of mould design and manufacturing stages by the use of InPerson. This was very helpful as they were able to

learn from the process and suggest modifications in a pro-active manner in situations where manufacturing process had to be repeated due to process defects.

Figure 5. Attributes gained by integrating the agent-based and CSCW approaches

Finally, the product development environment gained the attributes shown in Figure 8 by integrating these two collaborative approaches into a common environment and operating the constituents simultaneously. This does not cover all the attributes in a typical product development process in an industrial setting, however, this experiment did show how a product development process could benefit by such an integration of collaborative approaches.

7 CONCLUSIONS

The paper presented a study on the integration of software agent-based and CSCW approaches in a distributed product development environment. The study first examined the merits and limitations of software agents and CSCW approaches and presented a multiagent architecture integrating the two approaches, leading to a prototype environment. A simple experiment on distributed product development was carried out in this prototype environment. The experimental test-bed comprised of a critical mass of tools to study a simple product development process. The results indicate that the integration of the two approaches did lead to a rapid product development process that benefited from human teamwork and agent autonomy. In addition this also led to the reuse of non-networked facilities. The complexity of the stepped shaft design problem was sufficient to demonstrate the process of integration. However, additional work is required to evaluate their applicability in a typical industrial setting. The environment did not include automatic mechanisms for efficient negotiation of conflicts, a task left for human agents to handle. More experiments need to be conducted in other environments with different configurations and software agents with more capabilities such as belief and learning. This would further enhance the existing integrated collaborative environment.

8 REFERENCES

- Berker, I and Brown, D.C., 1996, Conflicts and Negotiation in Single Function Agent Based Design Systems, *CERA Journal*, **4**, pp. 17-33.
- Cutkosky, M.R., et al, 1996, Madefast: Collaborative Engineering over the Internet, *Communications of the ACM*, **39**, pp. 78-87.
- Chawdhry, P.K., 1994, *Interactive Modelling of the Design Process as a Finite State Machine*, The 10th ISPE/IFAC International Conference on CAD/CAM, Robotics and Factories of the Future, August 21-24, Ottawa, Canada.
- Deitz, D., 1997, Product Development on the Web, *Mechanical Engineering*, pp. 66-69.
- Erkes, J.W., et al, 1996, Implementing Shared Manufacturing Services on the World-Wide Web, *Communications of ACM*, **39**, pp. 34-45.
- Frost, H.R., and Cutkosky, M.R., 1996, Design for Manufacturability via Agent Interaction, *ASME Design for Manufacturing Conference, Irvine, CA, USA, August 18-22. (96-DETC/DFM-1302)*
- Hardwick, M., et al, 1996, Sharing Manufacturing Information in Virtual Enterprises, *Communication of ACM*, **39**, pp. 34-45.
- Ndumu, D.T., and Nwana, H.S., 1997, Research and development challenges for agent-based systems, *IEEE Proceedings on Software Engineering*, **144**, pp. 38-50.
- Norman, D., 1994, How Might People Interact with Agents, *Communication of ACM*, **37**, pp. 68-71.
- Sreeram, R.T., and Chawdhry, P.K., 1997, *A Java Agent for Communication in a Concurrent Distributed Product Development Environment*, CE-97, Detroit, MI, USA.
- Sreeram, R.T., and Chawdhry, P.K., 1998, Human-centred Integration of Stand-alone Manufacturing Facility in a Networked Product Development Environment, *Under review for journal publication*

9 BIOGRAPHY

R. T. Sreeram is currently pursuing Ph.D. degree in Mechanical Engineering at the University of Bath, United Kingdom. His research interests include collaboration technologies, enterprise integration, distributed product development and human-centred manufacturing systems. He also holds M.S. and B.Tech degrees in Mechanical Engineering.

Dr. P.K. Chawdhry has been a lecturer in Design and Concurrent Engineering at the University of Bath since 1993. His research interests are in collaborative product development environments, intelligent systems and network design tools. Previously he held research positions at the University of Newcastle and University of Leeds where he researched in product data modelling, design integration and solid modelling.