CHAPTER 14

COMPUTER SUPPORTED COLLABORATIVE DESIGN: TECHNOLOGIES, SYSTEMS, AND APPLICATIONS

WEIMING SHEN[1], JEAN-PAUL BARTHÈS[2], and JUNZHOU LUO[3]
[1]National Research Council Canada, Ottawa, ON, Canada
[2]UMR CNRS 7253 Heudiasyc, Université de Technologie de Compiègne, Compiègne Cedex, France
[3]School of Computer Science and Engineering, Southeast University, Nanjing, PR China

14.1 INTRODUCTION

Industries and societies today require new technologies to address increasingly complex design issues for products, structures, buildings, systems, processes, and services while meeting the high expectations of customers. Computer supported collaborative design (CSCD) has emerged in response to this requirement. With the rapid advancement of Internet and Web-based technologies, CSCD was a very active research and development (R&D) area in the past two decades and progressed dramatically. To achieve its full potential, more and more research and commercial CSCD systems have been recently developed. The depth and breadth of these systems and applications are far beyond the traditional definition of concurrent engineering [1]. CSCD is carried out not only among multidisciplinary design teams within the same company but also across the boundaries of companies and time zones, with increased numbers of customers and suppliers involved in the process. This chapter presents a literature review of this R&D area, particularly CSCD technologies, systems, and applications. Research challenges and opportunities on CSCD are also discussed and highlighted.

Contemporary Issues in Systems Science and Engineering, First Edition.
Edited by MengChu Zhou, Han-Xiong Li and Margot Weijnen.

14.2 HISTORY OF COMPUTER SUPPORTED COLLABORATIVE DESIGN

14.2.1 CSCD

Traditional engineering design systems use a sequential mode of design generation, which breaks a design task into a number of sub-tasks that can be sequentially executed in a predefined workflow. Recently, such a sequential design mode has been found to be brittle and inflexible. It often requires numerous iterations, which make design expensive and time-consuming, and also limits the number of design alternatives that can be examined. On the other hand, sequential design is usually practiced with a downstream-wise information flow. Information feedback from downstream operations to the upstream design is usually performed by human interactions. It may thus cause insufficient design evaluation and optimization.

CSCD, also called cooperative design, concurrent design, or interdisciplinary design, is the process of designing a product, structure, system, process, or service through collaboration among multidisciplinary teams associated with the entire lifecycle. In the context of mechanical engineering design, this includes those functions such as preliminary design, detailed design, manufacturing, assembly, testing, quality control, and product service as well as those from suppliers and customers [2]. An important objective of CSCD is to address the insufficient or even absent manufacturability checks concurrently by detecting and considering conflicts and constraints at earlier design stages. To support collaborative design, information and communication technologies are used to augment the capabilities of the individual specialists, and enhance the ability of collaborators to interact with each other and with computational resources. CSCD is not only compulsory for complex products such as the development of the Boeing 777 airplane, which involves 130,000 parts, 6,800 internal people, and more than 10,000 external people, but also quite helpful for many middle- or even small-size products such as tooling and electronic products [3].

With the globalization of the manufacturing industry, CSCD is required to support distributed design. Members on a collaborative team often work in parallel and independently using different engineering tools distributed at remote locations, even across enterprise boundaries and across various time zones around the world. The resulting design process is then called distributed collaborative design [4].

Engineering design has some unique characteristics, for example, diverse and complex forms of information, interdisciplinary collaboration, and heterogeneous software tools, which make interactions difficult to support. Traditional approaches to sharing design information among collaborators and their tools include the development of integrated tools and the establishment of common data standards. These approaches are not good at supporting effective collaborative design because of the highly distributed nature of the design teams and engineering tools as well as the complexity and dynamics of design environments. A successful implementation of CSCD needs: (1) a series of new strategies, including an efficient communication strategy for a multidisciplinary group of people from the design and manufacturing departments to share and exchange ideas and comments; (2) an integration strategy to

link heterogeneous software tools in product design, analysis, simulation, and manufacturing optimization to realize obstacle-free engineering information exchange and sharing; and (3) an interoperability strategy to manipulate downstream manufacturing applications as services to enable designers to evaluate manufacturability or assembleability as early as possible [5]. On the other hand, the objective of a design team has multiple facets, for example, optimizing the mechanical function of the product, minimizing the production or assembly costs, or ensuring that the product can be easily and economically serviced and maintained. Achieving global satisfaction, cooperative strategy, such as negotiation, optimization, and trade-off, is an important aspect of CSCD.

14.2.2 CSCD Eve: 1980s

This section provides an overview of the related research fields, including CSCW (computer supported cooperative work), concurrent engineering, and HCI (human–computer interaction), which triggered the emergence of CSCD.

14.2.2.1 Computer Supported Cooperative Work According to Schmidt and Bannon [6], the term *Computer Supported Cooperative Work* (CSCW) was first used by Greif and Cashman in 1984 to describe the topic of an interdisciplinary workshop that they were organizing on how to support people in their work arrangements with computers [7]. Subsequently, the term was abbreviated to CSCW. The definition of CSCW and the history of this research field are beyond the scope of this chapter. Readers are suggested to consult a well-established journal called *Computer Supported Cooperative Work* (CSCW). In fact, many people simply refer to this area by the term of *Groupware*, though others consider this to be too narrow. Generally speaking, *Groupware* is widely used in commercial software products, whereas CSCW is used more often in the research community.

14.2.2.2 Concurrent Engineering The concept of concurrent engineering was initially proposed in the late 1980s as a potential means to minimize product development time. It was defined as "a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support" [8]. In a concurrent engineering environment, techniques, algorithms, and software tools are connected to allow product designers and developers to interact with each other. With concurrent engineering, more time and money are usually spent in the initial design stage to ensure the overall optimization of concept selection. Product design changes can be reduced at the late stages, leading to better engineered products with better total quality, time, and cost competitiveness. There are a number of implementation strategies, from the parallelization of product lifecycle functions to the upfront consideration of DFX activities such as design for manufacturability, assembleability, serviceability, and recycleability, to the cooperation and coordination of product design teams with different expertise [8, 9], all of which have laid a solid foundation for CSCD. To ensure the success of concurrent engineering, more emphasis is put on the establishment of team-work culture between design and

manufacturing teams, and the enhancement of quick and effective communication. Balamuralikrishna *et al*. [10] summarized concurrent engineering as three T's: tools, training, and time. *Tools* refer to the communication facilities between the personnel in the multidisciplinary departments to address the information exchange that is obstructed by the complexity and wide range of specialized disciplinary areas and interdependent activities. *Training* provides a mechanism for employees to work collaboratively and concurrently, making the best use of the company's resources. *Time* means that corporations need time to carefully investigate and plan concurrent engineering as it involves many complex software tools and information infrastructures. Many reported cases have shown that a hurried implementation of concurrent engineering usually has a high probability of backfiring. In industry, more companies have realized the great benefits of concurrent engineering.

14.2.2.3 Human–Computer Interaction Research on human–computer interaction (HCI) was started as early as computers were invented. Myers [11] presented a brief HCI history. However, there is currently no widely agreed definition that covers a full range of topics that form the area of HCI, from computer graphics to ergonomics, and from virtual reality to digital human modeling.

Computer graphics was born from the use of CRT and pen devices very early in the history of computers. Work in computer graphics has continued to develop algorithms and hardware to allow the display and manipulation of ever more realistic-looking objects—which led to rapid developments of CAD/CAM tools in the 1980s.

There are many HCI-related international conferences with the most widely recognized one being the HCI International Conference Series [12], as well as scientific journals like *Human–Computer Interaction* [13]. Please refer to Reference 14 for more resources on the subject.

With its initial R&D focus on interaction between one user and one computer, HCI R&D was then extended to human–human interaction via networked computers, which is, in fact, the essence of CSCD.

14.2.2.4 Blackboard, DAI, and Software Agents The blackboard architecture was proposed in the HEARSAY project [15] as a means to organize and control large artificial intelligence (AI) systems. Its first version, HEARSAY I was used for speech recognition based on the idea of cooperating independent acoustic, lexical, syntactic, and semantic knowledge sources.

The introduction of the Contract-Net is a milestone in the history of distributed artificial intelligence (DAI). The Contract-Net protocol was developed by Smith [16] and demonstrated on a distributed sensing system. The Contract-Net implemented a negotiation-based approach for allocating tasks dynamically to nodes in the network. When a node has to solve a problem that is beyond its expertise, it broadcasts a task-announcement message that describes the task to be solved. Nodes that receive the message and wish to solve the problem then return a bid message. The node that issued the task-announcement message, called the *manager*, waits for bids for a certain period and then selects one (or more) bidder(s) to do the task, who is called the *contractor*. Thus, the choice of the contractor is done after the selection by the

manager and by mutual agreement. To be able to function correctly, the system must include a high-level protocol that defines several types of messages with a structured content. Contract-Net has been widely used in various agent systems for negotiation among agents.

The Contract-Net protocol offered an early practical means for dealing with open systems from a software engineering point of view. Contrary to the blackboard approach, there is no shared memory where data and partial results are made available to the various knowledge sources. The Contract-Net approach has separate knowledge sources attached to distinct nodes in a computation network.

The actor model proposed by Hewitt [17] offers a model of computation for open systems at a finer grain than the Contract-Net approach. In the actor approach, problem solving is viewed as the result of the interaction of the activities of knowledge sources working independently and locally (with limited expertise and limited knowledge). Each node communicates with a limited number of other knowledge sources.

The concept of agents has evolved from the concepts of blackboard, Contract-Net, and actors. Separately, in such applied fields as manufacturing, object-oriented systems were developed with increasing intelligence being incorporated into the objects. What began as passive objects became "active objects" or "rule-based objects" or "intelligent objects," and finally "intelligent agent objects" as this stream of evolution merged with that of DAI [18]. All these technologies provide a good foundation for developing collaborative design systems. Under the CSCD context, an agent can be considered as a software system that communicates and cooperates with other software systems to solve a complex problem that is beyond the capability of each individual software system [19].

14.2.3 CSCD Emergence: 1990s

Modern design, particularly engineering design, is intrinsically multidisciplinary [20]. Various tools such as CAD/CAM/CAE (Computer-Aided Design/Computer-Aided Manufacturing/Computer-Aided Engineering) tools, developed and commercialized by different vendors without common specifications (or even with intentionally defined unique specifications for self-protection), do not address the needs of multidisciplinary design. On the other hand, large organizations, for example, Boeing, Airbus, or GM, must find a way to coordinate their R&D teams that are geographically distributed around the world in an effective way to carry out new product developments within a very limited time frame. Technologies like CSCW and intelligent agents have been investigated to meet this need, particularly to enhance communication, cooperation, and coordination among design team members as well as software tools. Some CSCW tools, like groupware, were directly used to facilitate communication among engineers and designers. Software agents were used to integrate different engineering tools. Examples of early applications of software agents in collaborative design include PACT [21], DIDE [22], and SiFAs [23].

With its emergence around 1993, the Web was quickly applied in the development of collaborative design systems, particularly for geographically distributed designers

to share design documents. Along with the Web, a number of associated representation technologies have been developed, such as HTML (Hyper Text Mark-up Language), XML (eXtensible Mark-up Language), VRML (Virtual Reality Mark-up Language), to enable better cross-platform and cross-enterprise exchange of multimedia information and design models. In terms of system infrastructure, many early collaborative design systems were also developed using the Blackboard architecture and distributed object technologies, like CORBA and COM/DCOM.

14.2.4 CSCD: Today

During the past two decades, a large number of CSCD systems were developed and reported, especially on the applications of CSCW, Web, software agents, and recently Web Services, Semantic Web, and Computing Grids, for collaborative design. A few CAD vendors and other software firms also started developing and promoting collaborative design tools, for example, AutoDesk's Inventor™ [24], Buzzsaw™ [25], Streamline™ [26], ArchiCAD TeamWork™ [27], Creo/elements Direct (formerly CoCreate's OneSpace Solution) [28], Dassault Systèmes' ENOVIA (formerly Matrix PLM Platform) [29], or UGS' PLM solutions [30].

CSCD has been one of the most important applications of CSCW technologies. The most widely used CSCW techniques in collaborative design systems include *groupware* techniques for facilitating communication among design team members and *context awareness* techniques for enhancing coordination among team members.

Web-based and agent-based approaches were dominant in the implementation of CSCD technologies, systems, and applications during the past two decades, and will be further discussed in the following sections.

14.3 METHODS, TECHNIQUES, AND TECHNOLOGIES

14.3.1 Communication, Coordination, and Cooperation

Communication, coordination, and cooperation are three fundamental aspects of CSCD technologies. Communication enables sub-systems/software agents/human design team members to exchange information and to coordinate their activities. Communication is the most important way in which cooperation and coordination among sub-systems/agents/people takes place. A detailed discussion can be found in Reference 31.

14.3.1.1 Communication Communication is crucial in distributed collaborative design systems. Indeed, the complexity of building a single system has been traded for a reduced complexity of each sub-system or agent. However, the difficulty has been transferred in part to the level of communication and to the issue of combining the separately achieved sub-tasks through cooperation and coordination.

Communication can be implemented in very different ways, depending on the nature of the sub-systems or agents, the global architecture of the system, the timing

of the exchanges, or the number of receivers for a message. Indeed, communication can occur between humans and/or machines. It can be direct by means of messages or indirect by posting; it can be synchronous or asynchronous; and it can concern a single sub-system/agent or several sub-systems/agents.

Communication can be direct or indirect. In multi-agent systems communication is direct between agents by using explicit messages. In client/server-based systems, communication is indirect by writing to the blackboard (which, however, is equivalent to a multicast communication strategy in the case of partitioned blackboards; broadcast otherwise). In the latter case, the communication mechanisms can be quite simple, like the SHARED workspace [32] in the DICE project.

Communication can be synchronous or asynchronous. Indeed, people work together via computer network in different modes: (1) co-located synchronous mode—they work in the same location and at the same time; (2) distributed synchronous mode—work in different sites but at the same time; (3) co-located asynchronous mode—they work in the same location but at different times; (4) distributed asynchronous mode—they work in different locations and at different times. Each mode requires hardware and software support for effective and efficient operations. It is important for users to recognize the distinctiveness of each mode, since their protocols, network, and storage requirements vary. In large design projects, both synchronous and asynchronous communications are required. Synchronous communication can be useful for teleconferencing (e.g., for design review meetings) [33]. However, most of the work will be done through asynchronous interactions.

The overall architecture of a design system has a strong influence on the way communications are organized. In client/server-based systems, the blackboard architecture has a tendency to impose indirect, synchronous, and general (broadcasted) communication. In multi-agent systems on the other hand, we find mostly direct, asynchronous communications with a variety in the number of receivers (point-to-point, multicast, and broadcast). The nature of the agents is somewhat orthogonal to the architecture of the system. Thus, we have several kinds of systems: (1) systems for supporting the communication between human designers and facilitating their access to various tools; (2) systems having mostly automated software agents; (3) systems having both human designers and automated software agents where communication occurs among the designers, among the agents, and between designers and agents. In the second category, human designers are of course present. However, the corresponding system usually has a single user, and the communication between the user and the system is traditional.

In order to support communication it is necessary to develop protocols. There are two levels of protocols: one for securing the communication process, another for supporting the exchange of information. They are supported by distinct communication languages. Among the protocols of communication proposed or developed, there exist some simple protocols such as Contract-Net [16] with a number of variations. The protocols supporting the exchange of information include a proposition from Gaspar [34] offering a protocol with four types of messages, each corresponding to the communication modes among agents; a more complex protocol like SANP [35],

based on the results of linguistic researches; and KQML (Knowledge Query and Manipulation Language) [36].

14.3.1.2 Coordination According to Lesser and Corkill [37], "the objectives of the coordination process are to ensure that all necessary portions of the overall problem are included in the activities of at least one agent, that agents interact in a manner which permits their activities to be developed and integrated into an overall solution, that team members act in a purposeful and consistent manner, and that all of these objectives are achievable within the available computational and resource limitations."

Large design projects include complex processes distributed across time, involving many participants and functional perspectives. The design of a new commercial jet, for example, requires the integrated contribution of thousands of individuals spread over several continents and spanning possibly decades of time. Effective coordination is critical to the success of this cooperative process since the distributed activities are typically highly interdependent due to shared resources, input–output relationships, and so on. The sheer complexity of these interdependencies has begun to overwhelm traditional manual organizational schemes and paper-based coordination techniques, resulting in often huge rework costs, slowed schedules, and reduced product quality. As a result, while individual productivity may be high, the failure of existing coordination support practices and technologies has severe impact on the bottom line [38].

Coordination is central to the successful operation of CSCD systems, particularly to agent-based collaborative design systems whose complexity is high and stability is essential. Without coordination, a group of agents can quickly degenerate into a chaotic collection of individuals. The easiest way of ensuring coherent behavior would be to provide the group with an agent that has a wider perspective of the system. Such an agent then becomes the central controller of the system. This central controller could gather information from the agents in the group, create plans, and assign tasks to individual agents in order to ensure coherence. In the PACT project [21], the controller role is played by agents called facilitators, in CONDOR by collective agents [39], and in ABCDE by managers [40].

Using a single central controller for a large group of agents has an obvious problem. As the size of the group increases it becomes very difficult for the central controller to be informed of all the agents' beliefs and intentions. Also such a controller can become a severe communication bottleneck and would render the remaining components/agents unusable if it failed.

Based on the comprehensive review of the literature, we summarize a number of fundamental mechanisms and techniques that can be used to support multi-agent coordination [31]:

- *Mutual adjustment*: Agents share information and resources to achieve some common goal, by adjusting their behavior according to the behaviors of the other agents. In order to do so, agents usually need to exchange considerable information and make many adjustments. No agent has prior control over the

others and the decision-making is a joint process. It is the simplest form of coordination. Coordination in peer groups and in markets is usually by mutual adjustment.

- *Direct supervision*: One agent has some degree of control over others and can control the use of shared resources by the subordinate agents and may also prescribe certain aspects of their behavior. Such a supervision relationship is often established through mutual adjustment (e.g., following acceptance of employment or a contract).
- *Coordination by standardization*: The supervisor agent coordinates other agents through standardization, by establishing standard procedures for subordinate agents to deal with situations they encounter. In mutual adjustments, such standard procedures are implemented by acceptance. While in direct supervision, they are implemented through (mandatory) requests. Routine procedures in companies or computer programs are examples of coordination by standardization.
- *Mediated coordination*: A mediator acts as a facilitator (e.g., finding or delivering information), and as a broker (a "go-between") and advisor on resource negotiations or supervisor (exercising some degree of direct supervision). The first role is mandatory, but the others are optional. A mediator facilitates or brokers mutual adjustment among agents and may also use direct supervision.
- *Coordination by reactive behavior*: Agents react to particular stimuli (situations) with specific behaviors (actions). With appropriately selected or evolved stimuli–behavior groupings and distributions, system-level patterns of coordinated behavior emerge so as to contribute to the achievement of common or system goals.

There are also other mechanisms for multi-agent coordination described in the literature such as Markov tracking [41], coordination techniques dealing with commitments and conventions [42], and knowledge-based reasoning techniques [43].

14.3.1.3 Cooperation Collaborative engineering design projects, particularly concerning those large and complex products and structures such as automobiles, locomotives, and airplanes, require the cooperation of multidisciplinary design teams using multiple sophisticated and powerful engineering tools such as commercial CAD/CAE tools, engineering database systems, and knowledge-based systems. Individuals or groups of multidisciplinary design teams work in parallel and independently with various engineering tools located at different sites. At any moment, individual members may be working on different versions of a design or viewing the design from various perspectives (such as profitability, manufacturability, resource capability, and capacity) at various levels of details. Effective cooperation is imperative.

In general collaborative systems, the cooperation can be fully cooperative, partly cooperative, and antagonistic. However, in CSCD systems there is usually no antagonistic situation, that is, it is either fully cooperative (particularly within the same

company) or partly cooperative (usually among collaborative companies or stakeholders).

Fully cooperative sub-systems/agents that are able to resolve non-independent problems often pay a price in high communication costs. These sub-systems/agents may change their goals to suit the needs of other sub-systems/agents in order to ensure cohesion and coordination. Most real CSCD systems have a weak degree of cooperation, that is, they are partly cooperative, in order to reduce the communication cost and design time.

Fully cooperative agents exist in cooperative distributed problem solving (CDPS) systems where agents work together in a loosely coupled network to solve problems that are beyond their individual capabilities. In this network, each agent is capable of contributing to solve complex problems and can work independently, but the problems faced by the agents cannot be completed without full cooperation. This level of cooperation is necessary because no single agent has sufficient knowledge and resources to solve a given problem, although different agents might have the expertise to separately solve its different parts (i.e., sub-problems). CDPS agents cooperatively build a solution to a problem by using their local knowledge and resources to individually solve sub-problems, and then by integrating the solutions for these sub-problems into an overall solution. CDPS is particularly useful when distributed control becomes uncertain. The presence of uncertainties, such as incomplete and imprecise information, makes the need for full cooperation even more crucial. CDPS is also useful when each agent has specific knowledge and does not have a clear idea about what it might do or what information it might exchange with others. In this context, Smith's Contract-Net [16] provides a cooperative framework to minimize communication and allow load balancing and distributed control, while also maintaining coherent behavior.

While there are different cooperation methods [44] in the literature for general collaborative systems, cooperation methods widely used in CSCD systems include coordination as discussed above and conflict resolution through negotiation as described below.

14.3.2 Negotiation and Conflict Resolution

Conflicts occur in multidisciplinary collaborative design environments for two main reasons: individual participants in the cooperative design process lack the complete knowledge of global objectives necessary to ensure that their locally preferred solutions are in alignment with these higher-level objectives, and individual disciplines have individual ideas about what constitutes the best design. Even individual design requires trade-offs because of competing design criteria, such as safety, cost, and social acceptance, as well as requirements and specifications. The ability of human designers to avoid or minimize conflict through judicious trade-offs and other methods is one of their most valuable skills.

Conflict detection and resolution are especially difficult when the design task as well as the knowledge concerning competing factors are distributed among different actors with different perspectives. Methods like negotiation, hierarchical structures,

constraint weakening, creation of new variables, or user interaction, can be used for conflict resolution. It is also possible to combine several such methods within the same system.

Researchers have developed conflict resolution strategies for various types of conflicts: introducing an approach to conflict resolution based on classifying the conflict and mapping it to a specific strategy [45]; resolving conflicts in resource allocation among cooperative agents [46]; resolving conflicts via negotiation [47–49]; assisting the designer by using expert systems that offer criticisms and suggestions concerning design decisions in SNEAKERS [50].

Two interesting examples can be found in the project NEXT-LINK [51] and in the project SHARED-DRIM [52]. NEXT-LINK uses a simple and ubiquitous notion of constraint to represent all conflicts and provides a support for human-guided conflict resolution. Next-Link architecture describes all conflicts in terms of constraint violations, using the Redux ontology [121]. Redux describes an AI search-based ontology that maps directly onto an engineering task-based ontology. A constraint violation in NEXT-LINK simply means that some combination of assignments has been declared to be inconsistent. This is not a logical notion of consistency, but rather it is dependent upon the domain semantics. Redux, as a domain-independent agent, has no notion of such semantics. So, it does not insist that every constraint violation should be fixed immediately, nor does it draw any inference from such violation. What Redux does do, given a conflict among a set of assignments, is to determine the AND/OR set of decisions supporting the set, and then notifies each of the agents involved in the constraint violation.

On the contrary, SHARED-DRIM [52] uses an automatic conflict-resolution technology for resolving the known conflicts when solutions are available in its knowledgebase. In SHARED-DRIM, human design agents use the constructs provided by DRIM to make their recommendation. Every time they make a recommendation, the system searches to see if there exists a recommendation that introduces/modifies the same object in the database. If there is no such object, the system creates the object. However, if the object exists in the database, the system retrieves it for comparison. If an inconsistency is found, the system informs all interested participants about the changes made to the object. In order to provide the causes of conflict, the system must isolate the model entities needed to understand the cause of change. This process is achieved by the hierarchy division function. Hierarchy division selects the part of an intent hierarchy from which the conflict under consideration can be explained. After the system has reached the level of detail or abstraction in which a source for the conflict can be signaled, it begins to explore the validity of the recommendation that is related to the source. To do this, the system uses a function called consistency checking. By detecting inconsistencies in any object and providing the causes of conflict, the system instantiates a negotiation process between all involved professionals. In this negotiation process, the computer may take an active or passive role. An active role implies that the system has access to knowledge that prescribes how to relax a specific type of constraint. On the other hand, the computer may take a passive role by providing design agents only with information on the existence of the conflict and the decision which led to the problem. The human design agents then make the decisions.

We believe that it is not possible to conduct large complex design projects while enforcing consistency at all levels. Thus, conflicts are inherent to a design process. Consequently, conflict resolution is a domain that will see many developments in the future.

14.3.3 Ontology and Semantic Integration

Collaborative engineering design projects are complex, involving many different stakeholders, and require sharing knowledge independently from the use of information and communication technologies. Ontological engineering is a new discipline that allows descriptions of the world to be handled within human–human, human–machine, and machine–machine communications, irrespective of the particular computing environments in which these occur.

Ontologies are a key element of CSCD. They both define and organize all concepts being used in the collaborative design process at different levels. Several ontologies are usually involved in the design process: (1) a domain ontology modeling all concepts to be found in the application domain; (2) a process ontology modeling concepts related to the process itself. A domain ontology helps different specialists to translate among the various professional jargons concerning the product domain. It allows indexing project documents semantically for easy retrieval. It can also be used to organize knowledge elements when capitalizing knowledge [53]. It also allows querying knowledge, either internally or externally through semantic Web Services [54]. A process ontology can be used as a backbone to organize the design process and provide context during different design phases. Combination of a domain ontology and a process ontology can be used to support natural language interaction, for example, between a human and his/her personal assistant agent.

Finally, because of globalization, ontologies need to be multilingual to support interaction in multi-cultural environments.

Unfortunately, it is difficult to build ontologies, to maintain them, and to use them correctly. The current popular formats like OWL produce structural information. Reasoners like RACER, Pellet, FACT++, Flora, JENA, JESS, Bossam, or SPARQL are defined outside the ontologies, which limits their efficiency. A comparison between reasoners was done by Huang *et al*. [55]. Significant R&D efforts are still required in this area.

14.3.4 Personal Assistance and Human–Machine Interaction

In the early 1970s, minicomputer-based CAD systems were composed of workstations including a graphics screen, a keyboard, a graphics tablet and stylus, and sometimes a large digitizer. At the end of the 1970s, engineers gathered around such workstations to work together, discussing new design around 3D wire mesh mock ups of a target product, for example, organizing the piping network of a power plant heat exchanger or organizing cranes on a construction site. Input could be done by means of graphical commands drawn on the tablet. Such small groups of designers were involved in co-located cooperative design, using the system as a recording device.

Later on, networks were introduced and one could envision distributed cooperative design. Today both organizations exist. They have different requirements.

14.3.4.1 Co-located Design Co-located design involves several people working together in the same environment. The tools must: (1) support the design process; (2) record design decisions; and, (3) prepare the final documentation [56]. Project review meetings are of this type.

First, CSCW human–machine interaction (HMI) benefits from all hardware novelties that have appeared on the market or are developed in the laboratories, starting from traditional screen/keyboard/mouse/menus interaction devices. Thus, we have seen the development of wall screens, virtual reality devices (CAVE, helmets, goggles), haptic devices, interactive boards, graphic surfaces, vocal inputs, tablets, smart phones, and 3D screens, and there will be clearly more to come.

The use of constructive solid geometry (CSG) and the possibility of displaying the designed product through 3D digital mock ups (DMU) using wall panels increased the quality of interaction for complex products. One step further was brought by collaborative augmented reality, allowing users to manipulate 3D objects in a virtual environment [57]. Today wall-size screens are replaced by large interactive touch panels. If different devices have been tested to interact with wall screens like magnetic rings or bracelets for pointing at the wall, an interesting approach consists in using laser pointers [58]. Haptic devices have been introduced to manipulate virtual 3D objects by means of gloves providing force feedback [59]. Sound and voice interfaces are proposed, for example, in the CALO project that tested a vocal interaction used in meetings during which participant interventions are recorded and the system is able to produce a summary of the discussion at the end of the meeting [60]. Voice interfaces and natural language interaction can also be used to select tasks, triggering specific dialogs before launching a particular task [61, 62]. The availability of graphics surfaces (tables) now offers the possibility of having multi-user multi-touch interaction [63]. Interaction is done through gestures [64] that still need to be standardized.

In the last 20 years, the development of the multi-agent technology brought major changes both to the architecture of the CSCW environments and to the HMI. The concept of personal assistant emerged slowly and now can be specialized to personal design assistants acting as specific technical secretaries that help doing the tedious repetitive work [65]. In some cases, personal assistant agents can be embodied in talking avatars [66]. The concept of digital butler proposed by Negroponte [67], consisting of giving a personal assistant a staff of specialized agents, allows an easier expansion of the system by adding more specialized agents as needed [68].

All the above devices need to be organized in consistent systems, and proposals have been done to develop systems containing wall screens and touch surfaces like the Roomware® approach [69] setting up a room combining the different devices, or the TATIN-PIC project combining multi-user multi-touch surfaces with vocal input and personal devices like tablets or smart phones for supporting preliminary cooperative design [63]. An interesting initiative is the PAL program proposed by DARPA with the goal of organizing a command center room for preparing (designing) missions cooperatively [70].

Second, interaction devices must be integrated and used in the context of design, meaning that supporting software has to be developed to this end. This includes 3D interaction, multimodal interaction, virtual reality, avatars, personal assistants, and much more.

As mentioned previously, CSG has been used to build 3D mock ups, which can be used to index design knowledge by attaching information [71]. Multimodal interaction includes now gestures that are charted to produce standards [64]. Multi-agent technology has been introduced as a supporting architecture, in particular with the definition of personal design assistants [65, 72].

14.3.4.2 Distributed Cooperative Design Distributing cooperation brings additional requirements with respect to co-located activities, namely exchange of documents, organization of the work, for example, version control, addressing the problem of awareness or coordination.

Distributed cooperative design was first developed around groupware and in particular document management (DM) systems like Lotus Notes® [73], as sharing documents is a prerequisite to collaborative work. Many DM products are today available including open source ones like Alfresco [74], Nuxeo [75], Zotero [76], or others. Another important requirement, in particular when developing software, is version control, which called for approaches like CVS [77], SVN [78], Git [79], and many others. Some researchers claim that such traditional systems are becoming inadequate and should be replaced by systems allowing real time collaborative editing like Google Wave [80]. Another problem when people are not co-located is that of awareness. Awareness of what other people are doing remotely needs be enforced [81, 82]. Finally, coordination is also more difficult to achieve when teams are distributed. This can be improved by the use of design assistant agents [83].

14.3.5 Collaborative Workflows

Large organizations must find a way to coordinate their R&D teams (distributed geographically around the world) in an effective way to carry out new product/process/system developments within very limited time framework. This requires not only simple integration of engineering tools but the seamless integration of the way they do their job, the efficiency of data sharing and transfer, and the awareness and consistency of design changes. Ambiguity in the description of roles, responsibilities, interactions, and processes makes collaborative design very difficult. Change management and product data coordination are two challenging issues as well.

Therefore we need to find a feasible technology to tackle the following requirements:

- integration of heterogeneous engineering tools installed in different environments at different locations;
- integration of multidisciplinary teams that may transverse organization boundaries;

- integration of segments of business logics of different engineering and manufacturing activities such that they can function as a whole business;
- effective data management and data exchange standards; and
- change management and version control.

We believe that three major technologies make up the collaborative workflow technology:

(1) web-related technology, such as WWW, Web Services, and Web Semantics, for a convenient media to publish, share, and exchange information and for providing unit engineering and manufacturing services over the Internet;
(2) intelligent agent technology as a semantic "glue" for adaptive process coordination and providing active assistance to a multidisciplinary team; and
(3) process modeling and coordination technologies—workflow.

Web is a convenient media to publish and share information relevant to the spectrum of the design process, from concept generation and prototyping to virtual manufacturing and product realization. Web-based infrastructure has been used in a number of collaborative design systems. In most cases, it is primarily used by multidisciplinary team members as a medium to share design data/information/knowledge and in some cases for product data management and project management through integration of the Web with the related technologies.

Web technology alone, however, is not the solution to collaborative design systems, although it makes communication physically viable through a common network. In order to work on a distributed project effectively, remote engineers and designers need active helps to coordinate their efforts. This coordination may involve the translation of terminology among disciplines or locating/providing engineering services (e.g., finite element analysis). Agent technology provides support to enhance the performance of collaborative design systems [31]. In agent-based collaborative design systems, agents have mostly been used for supporting cooperation among designers, providing a "semantic glue" between traditional tools, or for allowing better simulations. Shen *et al*. [31] provided a detailed discussion on issues in developing agent-based collaborative design systems and a review of significant, related projects or systems.

Both agent technology and Web technology are very useful in the integration of design processes. The collaborative workflow strategy still requires for a third technology called workflow. Workflow management is a fast evolving technology which is increasingly being exploited by businesses in a variety of industries. Its primary characteristic is the automation of processes involving combinations of human- and machine-based activities, particularly those involving interaction with information technology (IT) applications and tools. In recent years, the need for reorganization, continuous improvement of business processes, and the advances in IT have created a huge market for workflow management systems. Modern workflow management systems enable the design, enactment, and monitoring of workflow in

a heterogeneous and distributed environment, allowing efficient process execution and management [84]. Very recently, Kim [85] presented a model-driven workflow fragmentation framework for collaborative workflow architectures and systems.

14.3.6 Collaborative Virtual Workspaces and Environments

With the integration of virtual reality, software agents, and Internet/Web-based technologies, collaborative virtual workspaces or virtual environments are being widely applied in almost all e-business and engineering domains for collaboration among distributed teams.

Rosenman *et al*. [86] presented a framework for collaborating in a virtual environment including an IFC-based database containing the various models and relationships, a virtual world environment for collaboration, and an agent-based society for handling communication among the users.

Aspin [87] proposed an interaction mechanism that enables a group of co-located users to collaboratively interact with a common visual environment through the use of lightweight remote computing devices. Applying an object-based distributed shared memory system enables the description of the active sessions to be distributed to both the collection of services, forming the design/review session configuration, and the remote interface applications that support individual user interaction. This distributed system then forms a synchronized, distributed description of the session content that both informs services of the session content and provides a centralized system for managing user interaction.

In an interesting experimental work, Hammond *et al*. [88] used a socio-technical theory as a framework to explore differences in engineering design team decision-making as a function of various communication media. Their results indicate that design teams communicating via an electronic medium perceive an increase in mental workload and interact less frequently, but for a greater total amount of time. These results brought interesting implications and suggestions for the management of distributed design teams or the management of human aspects [89].

14.3.7 New Representation Schemes for Collaborative Design

In a collaborative design process, product models need to be disseminated in a broader scope. Product models are the most important knowledge and properties of the product development companies. As a result, companies are usually reluctant to share these models directly to avoid the leakage of the commercial secrets to competitors. This consideration makes it difficult to realize the full potential and benefits of collaboration. On the other hand, a product model is proprietary to a CAD system. In a collaborative design environment with multiple users, it is infeasible or uneconomical to install a CAD system for every user to view or manipulate the product model. To address these concerns, research efforts have been made to develop new representation schemes of product models based on VRML, including X3D (eXtensible 3D) [90], Web 3D [91], U3D (Universal 3D) [92], JT [93], or OpenHSF [94]. These representation schemes retain the essential visualization information of

proprietary product models to support display-based manipulations, such as rotation and zooming, annotation, and mark-up. Most of these schemes are open in formats and the features inside are neutral such that they have much broader acceptance than those of the proprietary product models. Major applications of these schemes for collaboration include customer surveys of product concepts and initial models, high-level project reviews among management, development and service departments, sales promotion, e-documents (e.g., Acrobat 3D), sharing catalogs, and visualization functions in product data management/product lifecycle management (PDM/PLM) systems. Since only the visualization information is included in these schemes, crucial design information is protected.

14.3.8 New Visualization Systems for Collaborative Design

In order to support the new representation schemes, some new visualization systems have been developed, for example, Cimmetry Systems AutoVue [95], Actify SpinFire [96], SolidWorks eDrawing [97], RealityWave ConceptStation [98], and Autodesk Streamline [26]. The visualization-based platforms are cost-effective solutions to replace CAD systems to facilitate collaborative activities for various users. With new representation schemes and visualization systems, teams can collaborate more effectively, such as by taking on design discussions, reviewing new products, and conducting customer surveys to get design feedback as early as possible. This may overcome some drawbacks of proprietary CAD product models that hinder collaborative activities. A visualization-based collaborative system uses a two-tier or three-tier client/server architecture. Java Applet and Microsoft ActiveX technologies are widely used for developing Web-based or specialized clients. Core functions or services are implemented in Java Servlet or Microsoft .Net ASP at the server side to provide system support and maintenance [99–101]. Recently, Java3D has been widely used to enable visualization-based manipulations of 3D objects and scenes, for example, to build, render, and control 3D objects for Web-based collaboration [102, 103].

14.3.9 Product Data Management and Product Lifecycle Management Systems

Product data management (PDM) and product lifecycle management (PLM) systems have been adopted by industry to facilitate engineering design. Such systems promise that the "right information" is provided to the "right person" in the "right time" according to the "right order." Mainstream solutions include UGS TeamCenter [104], PTC Windchill [105], ENOVIA VPLM, ENOVIA MatrixOne, and ENOVIA SmarTeam [106]. Actually, the systems can be regarded as the system-level integrated implementation of the current collaborative technologies to support engineering design [107]. These systems have distinguished characteristics, while sharing the following common functionalities:

- team management—to map the structure of a product development team to a hierarchical structure of organizations;
- product structure management—usually a bill of materials (BOM) structure root, to represent the physical structure of a developed product at different levels, which generally contains assemblies, sub-assemblies, and components;
- workflow and process management—to allow an organization to automate procedures in which information, tasks, and documents are passed among participants;
- design change management—to manage change information in design processes;
- visualization-based collaborative workspace—to retain the visualization information of product models based on light-weight visualization schemes to support multiple users to manipulate the product models, such as rotation, measurement, annotation, and mark-up; and
- integration interfaces with CAD, shop floor execution systems, legacy enterprise information systems, and other partners on the product value chain.

From a research perspective, important issues need to be solved for the better application of PDM/PLM systems. The design process, along with the product itself, should be considered as a crucial component of an engineering enterprise's intellectual capital [108]. Five aspects of design processes have been studied, including support for design information transformations, support for design decision-making, modeling and representation of design processes, analysis of design processes, and synthesis of design processes. Qiu and Wong [109] developed a dynamic workflow mechanism to accommodate the changes during design by minimizing the repetitive execution of finished workflow nodes. This approach can address the data integrity issue by managing various workflow data such as node properties and scripts. Concurrency control is the foremost mechanism to organize synchronous activities to avoid conflicts. Locking is a primary means in managing concurrency control to prevent people from colliding, and three types of locking, that is, non-optimistic, optimistic, and visual, have been developed and used in various applications. Negotiation can formalize and implement the mediation and facilitation functions among people to handle conflicts. Some research projects [107, 110, 111] have been carried out to enhance PDM systems to support pre- and post-design stages. Huang *et al*. [112] developed a Web-based system to manage engineering changes.

14.3.10 Security and Privacy

A major concern of implementing network-enabled collaborative design systems is the assurance that proprietary information about the intellectual property owned by an organization or information about the company operations is available to authorized individuals only. Collaborative design involves sharing intellectual property in the form of detailed design information as well as competitive enterprise information. For

general acceptance of the collaborative design approach, the secrecy of the proprietary or competitive information must be maintained.

In addition to maintaining secrecy, collaborative design systems must also accommodate privacy of the individuals and organizations involved in collaborative design activities. Gathering and processing information about the activities of individuals or groups while managing or operating processes or machinery via computer networks can provide a great deal of details concerning the ways in which the individuals interact as well as process-related information. In a highly competitive business world, we must ensure that business intelligence information or the information provided by individuals or organizations is only shared in a fashion dictated by those involved.

14.4 COLLABORATIVE DESIGN SYSTEMS

14.4.1 System Architectures

In all kinds of collaborative design systems, two types of system architectures are dominant: the client/server architecture used mostly in web-based collaborative design systems as discussed in detail in Section 14.4.2, and federated system architectures used in most other collaborative design systems, including agent-based collaborative design systems and service-oriented collaborative design systems.

The client/server system architecture has been popular in traditional distributed computing systems, particularly web-based systems. A number of federated architectures have been proposed in the literature. Three approaches, facilitators, brokers, and mediators, are well known and widely used in collaborative design systems. The facilitator approach was proposed for the SHADE project [113] and demonstrated in the PACT project [21], where several facilitators (facilitation agents) were used to ease the communication and coordination among agents. CIIMPLEX [114] and some other agent-based collaborative design systems also use this approach. Brokers (also called broker agents) have been used in a number of agent-based collaborative design and manufacturing systems [115, 116]. The mediator approach was initially proposed by Wiederhold [117] for distributed database management systems. It has also been successfully used in collaborative design and manufacturing system [118].

In addition to the three federated approaches described above, there are some other agent-based collaborative engineering design and manufacturing systems that could be classified as federated architectures. The First-Link project [119] was intended to test a specific software architecture where a central product model (*central node*) is accessed by independent software modules, somewhat like a blackboard architecture. The Next-Link project [120] was the successor to First-Link and was to study how to coordinate the independent agents, with a mechanism called the *Redux'* services developed by Petrie [121] being added to the system for this purpose. In the Process-Link project [122], this type of *central node* was replaced by an *agent manager* which is very similar to the previously described mediator. SiFAs [23] used a similar architecture with a *design board* that is visible to all agents participating in design. Lander *et al*. [123] proposed using a multi-blackboard architecture for managing agent

interactions within a collaborative engineering design system. CAAD (Cooperative Agents and Applications in Design) used a similar approach to allow several designers to cooperate through a multi-blackboard system [124].

Other collaborative design system architectures include the autonomous agent approach in DIDE [22]. The DIDE project used this approach in implementing agent-based intelligent engineering design environments. DIDE is organized as a population of asynchronous cognitive agents integrating engineering tools and human specialists within an open environment. Each tool (or interface for a human specialist) is encapsulated as an agent. Engineering tools and human specialists are then connected through a local network and communicate via this network. Each can also communicate directly with other agents located in any other local networks, using the Internet. All agents are independent and autonomous. They exchange design data and knowledge via a local network or the Internet. DIDE does not use any facilitators or mediators. There is no static global control structure in the system. DIDE is specially designed to allow dynamic changes in the agent set (i.e., one can add new agents to or remove existing agents from the system without stopping and reinitializing the working environment). Some service-oriented collaborative design systems also use the autonomous agent approach [125–127].

One of the important features of autonomous agents is their independence. However, agents in federated multi-agent systems do not act completely independently. They communicate or interact with other agents through facilitators, brokers or other types of middle agents, and may be coordinated through agents such as mediators.

In the facilitator architecture, agents interact through facilitators that translate agent-specific knowledge into and out of a standard knowledge interchange language. Each agent can therefore reason (internally) in its own terms, asking other agents for information and providing other agents with information as needed through the facilitators. The facilitator model supports greater flexibility than direct communication used in the autonomous agent systems, because agents do not need to know detailed information about which other agents exist and what their capabilities are.

In the mediator approach, learning (where implemented) is at the group level, while in the autonomous agent approach, learning is at the individual level. Each autonomous agent has to have knowledge about its environment and the other agents, and has to learn in some sense if it is to update its knowledge.

Initially, one may think that the federation approach abandons the decentralization principle of distributed organizations by redefining centralized architectures. This is not so, since the ultimate goal of a federation organization is to release the agents from the burden of establishing their own communication links and assisting them to coordinate their activities with other agents. In such systems, individual agents are then free to concentrate their reasoning capabilities on autonomous planning and responding to the environment's stimuli. When clusters are formed, the organization at any instant may appear to be composed of "partial hierarchies." If these structures are dynamically created, however, and destroyed when their tasks are accomplished, it becomes evident that centralization is only being used locally, temporarily, and where it is advantageous.

In summary, the autonomous agent architecture is well suited for developing collaborative design systems when existing engineering tools are encapsulated as agents and connected to the system for providing special services, and the system consists of a small number of agents. Federated architectures are suitable for developing more complex collaborative design systems composed of a large number of problem-solving agents, data/information agents, and service agents. Implemented appropriately, these architectures can provide computational simplicity and manageability. The client/server approach will continue to be used for industrial applications, but they will be progressively replaced by the federated architectures and the autonomous system architectures.

14.4.2 Web-Based/Centralized Systems

The Web was originally developed for information sharing within internationally dispersed teams and the dissemination of information by support groups. Proposed and developed early in the 1990s, the Web has quickly become a convenient media to publish and share information relevant to the design process, from concept generation and prototyping to virtual manufacturing and product realization. It has been adopted as the most popular implementation architecture of a collaborative product development (including design and manufacturing) tool. A CSCD system developed with the Web as a backbone will primarily provide: (1) access to catalog and design information on components and sub-assemblies; (2) communication among multidisciplinary design team members (including customers, designers, and production engineers) in multimedia formats; and (3) authenticated access to design tools, services, and documents. However, since the Web is still fast evolving, particularly with the development of Web Services, Semantic Web, and Cloud Computing technologies, many researchers and working groups in and outside the World Wide Web Consortium are working hard to improve the current Web infrastructure and supporting tools. Web-based infrastructure has been used in a number of collaborative product design systems. In most cases, the Web is primarily used by multidisciplinary team members as a medium to share design data/information/knowledge; while in some cases, it is integrated with other related technologies and is used for product data management and project management.

A comprehensive review of some Web-based tools and systems can be found in References 4 and 128. Most Web-based collaborative design systems are developed using Java and CORBA [129–131], and some others are developed using Common Lisp, for example, WWDL [132], or Prolog, for example, WebCADET [133]. In addition to HTML and Java Applets for developing client-side user interfaces, ActiveX [134, 135] and VRML [131, 132] are widely used.

However, Web technology alone is not a complete solution to collaborative design systems, although it makes communication physically viable through a common network. In order to collaborate on a distributed design project, remote engineers and designers need active supports to coordinate their efforts. This coordination involves the translation of terminology among disciplines, locating/providing engineering analysis services, virtual prototyping services, and project management [136–138].

Web servers should not only be a repository of information but also provide intelligent services to help users to solve design problems. Such servers may be called software agents and will be discussed below.

14.4.3 Agent-Based/Distributed Systems

Application of software agents to collaborative design has been demonstrated by various research projects. PACT [21] might be one of the earliest successful projects in this area. The interesting aspects of PACT include its federation architecture using facilitators and wrappers for legacy system integration. SHARE [139] was concerned with developing open, heterogeneous, network-oriented environments for concurrent engineering, particularly for design information and data capturing and sharing through asynchronous communication. SiFAs [23] was intended to address the issues of patterns of interaction, communication, and conflict resolution using simple single-function agents. DIDE [22] was developed to study system openness, legacy systems integration, and distributed collaboration. ICM [140] developed a shared graphical modeling environment for collaborative design activities. Co-Designer [141] was intended to support localized design agents in the generation and management of conceptual design variants. Concept Database [142] described a strategic design support for version control, workflow management, and information gathering. A-Design [143] presented a new design generation methodology, which combines aspects of multi-objective optimization, multi-agent systems, and automated design synthesis. It provided designers with a new search strategy for the conceptual stages of product design that incorporates agent collaboration with an adaptive selection of design alternatives. Some projects also addressed the issue of integration and collaboration among product design, process planning, and manufacturing scheduling [118, 144, 145].

In agent-based collaborative design systems, software agents are mostly used for supporting cooperation among designers, enhancing interoperability between traditional computational tools, or allowing better simulations (particularly distributed simulations) [31].

14.4.4 Service-Oriented Systems

Both the Web and agent technologies are very useful in implementing collaborative design systems. The attractiveness of the Web for propagating information makes it appropriate to integrate with agents for accessing and manipulating information automatically. The challenge is to build a Web-based environment that enables and supports seamless interactions among human designers, software agents, and Web servers using the available emerging technologies [146].

A Web-based collaborative design system usually uses a client/server architecture in which the interaction between components is predefined and components are strongly coupled. This kind of approach is insufficient to support dynamic collaborative design environments, where tasks are usually involving complex and

nondeterministic interactions, producing results that might be ambiguous and incomplete. An agent-based collaborative design system is a loosely coupled network of problem-solvers that work together to solve problems that are beyond their individual capabilities [31]. Software agents in such systems are communicative, collaborative, autonomous (or semi-autonomous), reactive (or even proactive), and intelligent. Different system architectures have been proposed and used to implement agent-based systems, as discussed in Section 14.4.1.

Although agent technology has been recognized as a promising approach for developing collaborative design systems, those agents that have so far been implemented in various prototype and industrial applications are actually not very "intelligent." In this view, agent applications in the Web-based collaborative design field are still facing many challenging questions. WebBlow [146] is an interesting attempt on the integration of the Web and software agents in implementing a multidisciplinary design optimization environment [147, 148]. Before the emergence of Web Services, the concept of an active Web server was proposed to integrate the Web and agent technologies [149]. Since the active Web servers have very similar features of Web Services, it is natural for the further work to implement collaborative design systems using Web Services [125].

During the past decade, more and more collaborative design systems have been developed using Web Services as well as Semantic Web and Grid Computing techniques.

14.4.5 Collaborative Design Over Supply Chain (Virtual Enterprise)

Design of a complex product, system, or structure involves multiple stakeholders over the supply chain. Early involvement of suppliers in the design process can significantly reduce design changes during the manufacturing/construction process, and therefore reduce the cost and time. The collaboration in this scenario is different from the collaborative design with one company, since companies involved in such collaboration are usually competing at least for profits. In fact, companies may even be competing to survive in the increasingly competitive global market and sometimes may be getting into legal confrontations. One of the major concerns on collaborative design over the supply chain is the product information sharing [150]. Therefore, there is a co-existence of competition and cooperation over supply chains [151]. Even though collaborative design over the supply chain is not a new research topic, there has not been much progress in this area.

Related R&D work has been focused on enterprise collaboration, either vertically along a supply chain or horizontally among peers (even competitors). Virtual enterprise (VE) is one of the most important types of enterprise collaboration. It has the highest demand for sophisticated ICT technologies. A virtual enterprise can be defined as "a network of independent organizations that jointly form an entity committed to provide a product or service" [152, 153]. Thus, from the customer's perspective, as far as that product/service is concerned, these independent organizations, for all practical and operational purposes, are virtually acting as a

single entity/enterprise. A similar approach is used in MetaMorph II [118] where a hybrid agent-based mediator-centric architecture is used to integrate partners, suppliers, and customers dynamically with the lead enterprise through their respective mediators within a supply chain network via the Internet and Intranets. A detailed discussion on applications of agent technology to Virtual Enterprise can be found in Reference 154.

14.5 APPLICATIONS

Most CSCD applications mentioned above are related to product design in mechanical engineering [31], as did the CAD technologies. Almost all major companies producing mechanical products and systems are using CSCD technologies both internally with the companies and external over their supply chains. However, challenges still exist in terms of communication of design rationale, management of product data, collaboration and coordination of design teams, cultural and social aspects of design, information security and privacy, etc., which will be further discussed in the following section.

Due to the fragmented nature and adversarial behavior of the construction sector, the development and deployment of collaboration technologies and systems in architecture, engineering, and construction are behind the manufacturing sector [31]. Because the use of a single central repository to store the design information is not usually a viable option in such a unique industry [155], distributed loosely coupled integration and collaboration solutions using intelligent agents and Web Services technologies would be the most promising [156].

Recently, Building Information Modeling (BIM) [157] has been considered as an important enabling technology that drives the construction industry in improving productivity and efficiency. It combines some features of CAD, PDM, and PLM in mechanical engineering design. It can facilitate collaboration among stakeholders during the design, construction, and maintenance of buildings and facilities. It can also streamline the information integration throughout the lifecycle of buildings and facilities. A particular interesting application of BIM and collaboration technologies in architecture, engineering, and construction is change management [158]. However, BIM application in the construction industry is still at an early stage. According to Howard and Bjork [157], "the formal standards on BIM, such as the IFCs are complex and have not had the resources for rapid development and promotion that their potential deserved," and therefore it will take some time for this approach to be widely adopted.

While the focus of this chapter is on CSCD applications in the manufacturing and construction industries, CSCD technologies have been applied to many other domains including aerospace, automotive, logistics, transportation, power and energy, healthcare, infrastructure, administration, social networks, entertainment, and most recently to emergence response.

14.6 RESEARCH CHALLENGES AND OPPORTUNITIES

CSCD has been an active R&D area for about two decades. Some manufacturing and engineering companies have partially implemented in-house collaborative design systems. We expect a great future for CSCD and envision future ideal collaborative design systems as being:

- fully integrated with all necessary software tools connected through the network covering the full product lifecycle from conceptual design to detailed design (with detailed modeling, simulation, and optimization), virtual prototyping, manufacturing/construction, service and maintenance, and final disposal;
- integrated with physical testing and validation systems for "hardware-in-the-loop" simulations during the new product/process/system/service development;
- implemented as semi-automated interactive systems that involve human interventions;
- operated on a collaborative computing environment with automated computing load balancing, quick access, and fast transfer of large volumes of engineering data;
- secured with sophisticated security and privacy protection mechanisms;
- able to allow users to choose favorite software tools according to their experience and preference;
- able to provide different users (including engineers and managers, sales and services staff, as well as customers and suppliers) with different access privileges to the same data/information;
- able to manage knowledge gained during previous design projects;
- able to advise users about good practices, taking into account their personal profile; and
- able to display a proactive behavior.

To achieve this vision, a number of challenging issues have been identified for an academic research, further development, and wider deployment of collaborative design systems in industry. In fact, these challenges are also opportunities for the CSCD research community. Among others, the following areas are believed to be future research opportunities and challenges:

- Ontology and semantics-based integration: One of the most difficult tasks in collaborative design is to agree on the ontological commitments that enable knowledge-level communication among the distributed design parties. Another difficulty is the integration of the various available design tools. If the tool data and models are encapsulated, rather than using a standardized and unified formalism, each tool will be free to use the most appropriate internal representations and models for its intended tasks. This is not a new research topic, but the

progress in this area has not been satisfactory. The emergence of the Semantic Web makes progress in this area more likely to occur.

- Interoperability among systems and/or models: Models help designers understand the nature of a design process by ignoring some of the not-so-important details. When deciding how to model a design process, determining the appropriate levels of abstraction is very critical for the model to be beneficial to its users. A key issue in collaborative design from a designer's perspective is how to bridge the multi-faceted models required to support a complex design project at various stages of the design process. The challenge is to use the relevant model for each task (the right abstraction and granularity) and to communicate the results in a suitable form to the various parties involved, whose needs are different and interests are diverse. One way to address this issue is through *collaborative design process modeling* which has been an active research topic recently, but significant efforts are still required.
- Reversibility and version control: A requirement is to record changes in a consistent manner through the various representations of the design product. This implies an extended form of version control as well as the possibility, may be limited, to propagate changes forward and backward. Also because the product is the result of changes that can be done by several groups in parallel, a reconciliation mechanism should be provided to integrate all changes at a given point.
- Product-centric design methodology: A product-centric design methodology is considered as a suitable approach for distributed collaborative product design. Featuring its self-learning ability, product-centric design fits well in a dynamically changing environment. Comprehensive care needs to be taken in modeling, collaboration, design, and development issues in the whole product lifecycle. Fundamental research is still required in this area.
- Data/information/knowledge management: Challenges in this area include knowledge discovery, support for natural language processing and information retrieval, the capturing of design intent in multimedia formats, dynamic knowledge management, self-learning, reasoning, and knowledge reuse. Based on the current Web infrastructure, users are allowed to access server resources primarily through HTTP and FTP. Using appropriate protocols to access the right data at the right locations is essential in collaborative design environments. This feature is particularly useful in large collaborative design and engineering projects where access to large volumes of data at different locations is frequent.
- Collaborative intelligent user interfaces: Human involvement in the collaborative engineering design processes is unavoidable. Designers need to interact with a design system and negotiate with peers via a user interface. The challenge is to make intelligent interfaces available to all resources such that the designers will have more flexibility for efficient and effective designing. The interfaces should be integrated, expressive, goal oriented, cooperative, easy to use, and customizable. In practice, the design process should be supported by a design environment in which the user is part of the system, rather than by a set of tools

for which the user is external to the system. This approach integrates the users' expertise and knowledge directly into the supporting design system.

- Distributed design project management: There must be some ways of managing all the resources involved, including people, organizations, software tools, and equipment. Relevant research issues are collaborative workflow, conflict management, cost and task management, activity scheduling, and computing resource management. In an interesting experimental work, Hammond *et al.* [88] used a socio-technical theory as a framework to explore differences in engineering design team decision-making as a function of various media of communication. Their results indicate that design teams communicating via an electronic medium perceive an increase in mental workload and interact less frequently, but for a greater total amount of time. These results brought interesting implications and suggestions for the management of distributed design teams. More research efforts are needed on how to increase people awareness when working in a distributed system.
- Drag and drop functionality: Drag and drop is a highly desired function in collaborative design using multiple computational tools. For example, a part designed under a CAD system may be moved to a CAE tool's graphical interface for analysis and simulation, and to a DFM tool's graphical interface for manufacturability analysis. It becomes more convenient if there is a drag and drop function that can copy or move a graphical object from one CAD/CAE system to another, particularly in a Web-based collaborative design environment. In fact, it is a type of communication between the two systems through the moving graphical object. The challenge is therefore to develop a common model or language for these related systems. A significant amount of research is needed to determinate standard geometric representations for features that can be used by different CAD and simulation tools. It also requires R&D of drag and drop type standards similar to OLE (object linking and embedding) which provides a protocol for organizing data in a standard format for exchange between different systems.
- Security and privacy: With the implementation and deployment of CSCD applications in industry, security and privacy issues become more and more important. The number of papers on this topic submitted to CSCWD conferences has increased significantly during the past few years. This will continue, particularly with more practical techniques and applications.
- Software self-management and self-healing: Since software self-management and self-healing have become an active research area, it would be natural to extend the research into CSCD systems.
- Social software and mass collaboration: Social software approaches and Wiki-style collaboration tools may be developed and used for knowledge-intensive collaborative design systems [159, 160].
- Cultural and social issues: With the industry globalization and the development of worldwide consortia, special attention is required with respect to cultural problems. Future collaborative design systems will need to integrate results

from social sciences in order to address the cultural differences of designers and users as well as be able to provide instant translation of natural language interactions.

- Intelligent knowledgeable environments should be developed in which a user is advised proactively with respect to the design process expertise, the domain expertise, or the knowledge acquired from previous projects. Such environments should be capable of acquiring knowledge on the fly and organizing it for future projects.
- Most systems are currently developed by engineers with the goal of filling technical needs. Usability studies need to be done extensively to make sure that the newly developed systems are easier to use and do not increase the cognitive load of the users. In general, human sciences have an increasingly important role to play in the future of CSCD.

14.7 CONCLUSIONS

CSCD has been recognized by industry as a way to address the requirements resulting from increasingly complex design of products, structures, buildings, systems, processes, and services, as well as to meet high customer expectations. With the fast development and advancement of Internet and Web-based technologies during the past two decades, various CSCD technologies and systems have been developed and applied to different domains including aerospace, automotive, manufacturing, logistics, transportation, power and energy, healthcare, infrastructure, administration, social networks, and entertainment. To achieve their full potential and the vision of fully integrated collaborative design systems, significant R&D efforts are still required. Some research challenges discussed above should be addressed within the next few years, though some of them may need a few decades to be thoroughly addressed.

REFERENCES

[1] J. Hartley. *Concurrent Engineering*. Cambridge, MA: Productivity Press, 1992.

[2] E. Sprow. Chrysler's concurrent engineering challenge. *Manufacturing Engineering*, 108(4): 35–42, 1992.

[3] K. T. Ulrich and S. D. Eppinger. *Product Design and Development*, 3rd edn. NewYork: McGraw-Hill, 2000.

[4] W. Shen and L. Wang. Web-based and agent-based approaches for collaborative product design: an overview. *International Journal of Computer Applications in Technology*, 16(2/3): 103–112, 2003.

[5] W. D. Li, S. K. Ong, and A. Y. C. Nee. *Integrated and Collaborative Product Development Environment—Technologies and Implementation*. World Scientific, Singapore, 2006.

[6] K. Schmidt and L. Bannon. Taking CSCW seriously. *Computer Supported Cooperative Work*, 1(1/2): 7–40, 1992.

[7] I. Greif (ed.). *Computer-Supported Cooperative Work: A Book of Readings*. San Mateo, CA: Morgan Kaufmann Publishers, 1988.

[8] J. Turino. *Managing Concurrent Engineering*. New York: Van Nostrand Reinhold, 1992.

[9] B. Prasad. *Concurrent Engineering Fundamentals: Integrated Product Development*, Vol. 2. One Saddle River, NJ: Prentice Hall, 1997.

[10] R. Balamuralikrishna, R. Athinarayanan, and X. S. Song. The relevance of concurrent engineering in industrial technology programs. *Journal of Industrial Technology*, 16(3): 1–5, 2000.

[11] B. A. Myers. A brief history of human computer interaction technology. *ACM Interactions*, 5(2): 44–54, 1998.

[12] HCI International Conference. Available at www.hci-international.org/ (accessed October 5, 2012).

[13] Human–Computer Interaction. Available at http://www.tandfonline.com/loi/hhci20 (accessed October 5, 2012).

[14] HCI Bibliography: Human–Computer Interaction Resources. Available at hcibib.org/ (accessed October 5, 2012).

[15] R. Reddy, L. Erman, R. Fennel, and R. Neely. The HEARSAY speech understanding system: an example of the recognition process. *IEEE Transactions on Computers*, C-25: 427–431, 1976.

[16] R. G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12): 1104–1113, 1980.

[17] C. Hewitt. Control structure as patterns of passing messages. In: *Artificial Intelligence: An MIT Perspective*, edited by P. H. Winston and R. H. Brown. The MIT Press, Cambridge, MA, USA, 1979, pp. 435–465.

[18] D. H. Norrie and A. D. Kwok. Object-oriented distributed artificial intelligence. In: *New Results and New Trends in Computer Science*, *LNCS 555*, edited by H. Maurer. Springer-Verlag, Berlin, Germany, 1991, pp. 225–242.

[19] W. Shen, Q. Hao, H. Yoon, and D. H. Norrie. Applications of agent systems in intelligent manufacturing: an update review. *International Journal of Advanced Engineering Informatics*, 20(4): 415–431, 2006.

[20] T. Tomiyama. Collaborative product development in ill-structured problem domains. In: Proceedings of the 10th International Conference on CSCW in Design, 2006, pp. 15–20.

[21] M. R. Cutkosky, R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum, and J. C. Weber. PACT: an experiment in integrating concurrent engineering systems. *IEEE Computer*, 26(1): 28–37, 1993.

[22] W. Shen and J. P. Barthès. DIDE: a multi-agent environment for engineering design. In: Proceedings of First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, CA, 1995, pp. 344–351.

[23] D. C. Brown, B. Dunskus, D. L. Grecu, and I. Berker. SINE: support for single function agents. In: Proceedings of Applications of AI in Engineering, Udine, Italy, 1995.

[24] Autodesk Inventor. Available at http://usa.autodesk.com/autodesk-inventor/ (accessed October 5, 2012).

[25] Autodesk Buzzsaw. Available at http://usa.autodesk.com/buzzsaw/ (accessed October 5, 2012).

[26] Autodesk Streamline. Available at http://usa.autodesk.com/adsk/servlet/pc/index?id=2164339&siteID=123112 (accessed October 5, 2012).

[27] Graphisoft ArchiCAD TeamWork™. Available at http://www.graphisoft.com/products/archicad/teamwork/ (accessed October 5, 2012).

[28] CoCreate, OneSpace. Available at www.cocreate.com/ (accessed October 5, 2012).

[29] Matrix PLM Platform. Available at http://www.3ds.com/products/enovia (accessed October 5, 2012).

[30] UGS, PLM Solutions. Available at http://www.ugs.com/solutions/ (accessed October 5, 2012).

[31] W. Shen, D. H. Norrie, and J. P. Barthes. *Multi-agent Systems for Concurrent Intelligent Design and Manufacturing*. London, UK: Taylor and Francis, 2001.

[32] A. Wong and D. Sriram. SHARED: an information model for cooperative product development. *Research in Engineering Design*, 5(1): 21–39, 1993.

[33] K. C. Lee, W. H. MansfieldJr., and A. P. Sheth. A framework for controlling cooperative agents. *IEEE Computer*, 26(7): 8–16, 1993.

[34] G. Gaspar. Communication and belief changes in a society of agents, towards a formal model of autonomous agents. *Decentralized Artificial Intelligence*, 2: 71–88, 1991.

[35] M. K. Chang and C. C. Woo. Speech-act-based negotiation protocol: design, implementation, and test use. *ACM Transactions on Information Systems*, 12(4): 360–382, 1991.

[36] T. Finin, D. McKay, and R. Fritzson. Specification of the KQML: agent-communication language. Technical Report EIT TR 92-04, Enterprise Integration Technologies, Palo Alto, CA, 1992.

[37] V. Lesser and D. Corkill. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, 11(1): 81–96, 1981.

[38] M. Klein. iDCSS: integrating workflow, conflict and rationale-based concurrent engineering coordination technologies. *Concurrent Engineering: Research and Application*, 3(1): 21–27, 1995.

[39] C. Iffenecker. Un système multi-agents pour le support des activités de conception de produits. Thèse de l'Université Paris VI, 1994.

[40] S. Balasubramanian and D. H. Norrie. A multi-agent intelligent design system integrating manufacturing and shop-floor control. In: *Proceedings of First International Conference on Multi-agent Systems*. The AAAI Press/The MIT Press, Cambridge, MA, USA, 1995, pp. 3–9.

[41] R. Washington. Markov tracking for agent coordination. In: *Proceedings of the Second International Conference on Autonomous Agents*. The ACM Press, New York, NY, USA, 1998, pp. 70–77.

[42] N. R. Jennings. Coordination techniques for distributed artificial intelligence. In: *Foundations of DAI*, edited by G. M. P. O'Hare and N. R. Jennings. John Wiley & Sons, New York, NY, USA, 1996, pp. 187–210.

[43] M. S. Mazer. Reasoning about knowledge to understand distributed AI systems. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6): 1333–1346, 1991.

[44] J. Ferber. *Les Systèmes Multi-Agents*. Paris: InterEditions, 1995.

[45] M. Klein. Supporting conflict resolution in cooperative design systems. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6): 1379–1390, 1991.

[46] S. E. Conry, R. A. Meyer, and V. R. Lesser. Multistage negotiation in distributed planning. In: *Readings in Distributed Artificial Intelligence*, edited by A. H. Bond and L. Gasser. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1988, pp. 367–383.

[47] K. P. Sycara. Cooperative negotiation in concurrent engineering design. In: *Computer-Aided Cooperative Product Development*, LNCS 492. Springer-Verlag, Berlin, Germany, 1991, pp. 269–297.

[48] B. Dunskus, D. L. Grecu, D. C. Brown, and I. Berker. Using single function agents to investigate negotiation. *AIEDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 9(4): 299–312, 1995.

[49] D. Bahler, C. Dupont, and J. Bowen. Anaxiomatic approach that supports negotiated resolution of design conflicts in concurrent engineering. In: *Artificial Intelligence in Design*, edited by J. S. Gero and D. Sudweeks. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 363–379.

[50] R. E. Douglas, D. C. Brown, and D. C. Znger. A concurrent engineering demonstration and training system for engineers and managers. *International Journal of CAD/CAM and Computer Graphics*, 8(3): 263–301, 1993.

[51] C. Petrie, M. Cutkosky, T. Webster, A. Conru, and H. Park. Next-Link: an experiment in coordination of distributed agents. In: Proceedings of AID-94 Workshop on Conflict Resolution, Lausanne, Switzerland, 1994.

[52] F. Pena, D. Sriram, and R. Logcher. SHARED-DRIMS: SHARED design recommendation—intent management system. In: Proceedings of 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE), 1993, pp. 213–221.

[53] C. A. Tacla and J-P. A. Barthès. A multi-agent architecture for evolving memories. In: AMKM 2003: Agent-Mediated Knowledge Management: LNCS 2926, 2004, pp. 388–404.

[54] M. L. Sbodio, D. Martin, and C. Moulin. Discovering semantic web services using SPARQL and intelligent agents. *Web Semantics Science Services and Agents on the World Wide Web*, 8(4): 310–328, 2010.

[55] T. Huang, W. Li, and C. Yang. Comparison of ontology reasoners: Racer, Pellet, Fact++. In: American Geophysical Union, Fall Meeting, 2008.

[56] C. A. Tacla, A. R. Freddo, E. C. Paraiso, M. P. Ramos, and G. Y. Sato. Supporting small teams in cooperative building application domain models. *Expert System Applications*, 38(2): 1160–1170, 2011.

[57] A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, and C. Bescher. Enveloping users and computers in a collaborative 3D augmented reality. In: Proceedings of IWAR'99 (International Workshop on Augmented Reality), San Francisco, CA, 1999, pp. 35–44.

[58] M. Wissen. Implementation of a laser-based interaction technique for projection screens. *ERCIM News*, 46: 31–32, 2001.

[59] I. A. Twombly, K. Montgomery, J. Smith, and R. Boyle. The virtual glovebox (VGX): a semi-immersive virtual environments for training astronauts in life science experiments. *Journal of Systemics, Cybernetics and Informatics*, 2(3): 30–34, 2004.

[60] G. Tur, A. Stolcke, L. Voss, S. Peters, D. Hakkani-Tur, J. Dowding, B. Favre, R. Fernandez, M. Frampton, M. Frandsen, C. Frederickson, M. Graciarena, D. Kintzing, K. Leveque, S. Mason, J. Niekrasz, M. Purver, K. Riedhammer, E. Shriberg, J. Tein, D. Vergyri, and F. Yang. The CALO meeting assistant system. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6): 1601–1611, 2010.

[61] A. Nguyen. An agent-based approach to dialogue management in personal assistants. PhD Dissertation, University of New South Wales, Australia, 2007.

[62] J.-P. A. Barthès. Flexible communication based on linguistic and ontological cues. In: Proceedings of the 5th International MCETECH Conference on eTechnologies, Les Diablerets, Switzerland, 2011, pp. 131–145.

[63] A. Jones, A. Kendira, D. Lenne, T. Gidel, and C. Moulin. The TATIN-PIC project: a multi-modal collaborative work environment for preliminary design. In: Proceedings of 15th International Conference on Computer Supported Cooperative Work in Design—CSCWD 2011, Lausanne, Switzerland, 2011, pp. 154–161.

[64] M. R. Morris, J. O. Wobbrock, and A. D. Wilson. Understanding users' preferences for surface gestures. In: Proceedings of Graphics Interface (GI'10), Ottawa, Ontario, 2010, pp. 261–268.

[65] K. Sugawara, S. Ben Yaala, Y. Manabe, C. Moulin, N. Shiratori, and J.-P. A. Barthes. Conversation-based support for requirement definition by a personal design assistant. In: Proceedings of 10th IEEE International Conference on Cognitive Informatics & Cognitive Computing, 2011, pp. 262–267.

[66] E. C. Paraiso, Y. Campbell, and C. A. Tacla. Webanima: a web based embodied conversational assistant to interface users with multi-agent based CSCW application. In: Proceedings of CSCWD 2008, 2008, pp. 337–342.

[67] N. Negroponte. *Being Digital*. Westminister, MD: Alfred a Knopf, Inc., 1995.

[68] J.-P. A. Barthès. OMAS—a flexible multi-agent environment for CSCWD. *Future Generation Computer Systems*, 27(1): 78–87, 2011.

[69] T. Prante, N. Streitz, and P. Tandler. Roomware: computers disappear and interaction evolves. *Computer*, 37(12): 47–54, 2004.

[70] DARPA PAL Program. Available at https://pal.sri.com/Plone/framework (accessed October 5, 2012).

[71] Thouvenin, I., Lenne, D., Guenand, A., and Aubry, S. Knowledge integration in early design stages for collaboration on a virtual mock up. In: Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design, vol. 2, 2005, pp. 1141–1145.

[72] Y. Zhang, H. Ghenniwa, and W. Shen. Agent-based personal assistance in collaborative design environments. In: *CSCW in Design II*, LNCS 3865, edited by W. Shen, K. M. Chao, Z. Lin, J. P. Barthes, and A. James. Springer-Verlag, Berlin, Germany, 2005, pp. 284–293.

[73] IBM Lotus Software. Available at http://www-01.ibm.com/software/lotus/ (accessed October 5, 2012).

[74] Alfresco. Available at www.alfresco.com/ (accessed October 5, 2012).

[75] Nuxeo. Available at http://www.nuxeo.com/en (accessed October 5, 2012).

[76] Zotero. Available at www.zotero.org/ (accessed October 5, 2012).

[77] Concurrent Versions System. Available at http://cvs.nongnu.org/ (accessed October 5, 2012).

[78] Apache Subversion. Available at http://subversion.apache.org/ (accessed October 5, 2012).

[79] Git. Available at http://git-scm.com/ (accessed October 5, 2012).

[80] Google Wave. Available at http://wave.google.com/ (accessed October 5, 2012).

[81] M. Kirsch-Pinheiro, J. Valdeni de Lima, and M. R. S. Borges. A framework for awareness support in groupware systems. In: Proceedings of CSCWD 2002, 2002, pp. 13–18.

[82] M. P. Locatelli and G. Vizzari. Awareness in collaborative ubiquitous environments: the multilayered multi-agent situated system approach. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(4), 2007, Article no.13.

[83] F. Enembreck, I. Thouvenin, M.-H. Abel, and J.-P. A. Barthes. An ontology-based multi-agent environment to improve collaborative design. In: Proceedings of 6th International Conference on the Design of Cooperative Systems, Coop04, French Riviera, France, 2004, pp. 81–89.

[84] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolution. *Data & Knowledge Engineering*, 24: 211–238, 1998.

[85] K. Kim. A model-driven workflow fragmentation framework for collaborative workflow architectures and systems. *Journal of Network and Computer Applications*, 35(1): 97–110, 2012.

[86] M. A. Rosenman, G. Smith, M. L. Maher, L. Ding, and D. Marchant. Multidisciplinary collaborative design in virtual environments. *Automation in Construction*, 16(1): 37–44, 2007.

[87] R. Aspin. Supporting collaboration, in co-located 3D visualization, through the use of remote personal interfaces. *Journal of Computing in Civil Engineering*, 21(6): 393–401, 2007.

[88] J. M. Hammond, C. M. Harvey, R. J. Koubek, W. D. Compton, and A. Darisipudi. Distributed collaborative design teams: media effects on design processes. *International Journal of Human–Computer Interaction*, 18(2): 145–165, 2005.

[89] T. Hartmann, M. Fischer, and J. Haymaker. Implementing information systems with project teams using ethnographic–action research. *Advanced Engineering Informatics*, 23(1): 57–67, 2009.

[90] X3D (eXtensible 3D). Available at www.x3d.com (accessed October 5, 2012).

[91] W3D (Web 3D). Available at www.macromedia.com (accessed October 5, 2012).

[92] U3D. Available at www.intel.com/technology/systems/u3d/ (accessed October 5, 2012).

[93] JT Open. Available at www.jtopen.com (accessed October 5, 2012).

[94] OpenHSF. Available at www.openhsf.org (accessed October 5, 2012).

[95] Cimmetry AutoVue. Available at www.cimmetry.com/ (accessed October 5, 2012).

[96] Spinfire. Available at http://autoweb.net/web2006/products-spinfire.shtml (accessed October 5, 2012).

[97] Solidworks eDrawings. Available at http://www.solidworks.com/edrawings (accessed October 5, 2012).

[98] RealityWave ConceptStation. Available at http://products.datamation.com/e-business/groupware/982097698.html (accessed October 5, 2012).

[99] N. Shyamsundar and R. Gadh. Collaborative virtual prototyping of product assemblies over the internet. *Computer-Aided Design*, 34(10): 755–768, 2002.

[100] L. Chen, Z. J. Song, and L. Feng. Internet-based real-time collaborative assembly modelling via an e-assembly system: status and promise. *Computer-Aided Design*, 36(9): 835–847, 2004.

[101] S. S. Zhang, W. Shen, and H. Ghenniwa. A review of internet-based product information sharing and visualization. *Computers in Industry*, 54(1): 1–15, 2004.

[102] L. Wang, S. Lang, and W. Shen. A Java3D enabled cyber workspace. *Communication of the ACM*, 45(11): 45–49, 2002.

[103] W. D. Li, J. H. Fuh, and Y. S. Wong. An Internet-enabled integrated system for co-design and concurrent engineering. *Computers in Industry*, 55(1): 87–103, 2004.

[104] UGS TeamCenter. Available at http://wwwprd.ugs.com/products/teamcenter/ (accessed October 5, 2012).

[105] PTC Windchill. Available at http://www.ptc.com/appserver/mkt/products/home.jsp?k=37 (accessed October 5, 2012).

[106] ENOVIA VPLM, MatrixOne and SmarTeam. Available at http://www.3ds.com/products-solutions/plm-solutions/enovia/products/ (accessed October 5, 2012).

[107] X. W. Xu and T. Liu. A web-enabled PDM system in a collaborative design environment. *Robotics and Computer Integrated Manufacturing*, 19(4): 315–328, 2003.

[108] J. H. Panchal, M. G. Fernández, C. J. J. Paredis, J. K. Allen, and F. Mistree. Leveraging design process related intellectual capital—a key to enhancing enterprise agility. In: *Collaborative Product Design and Manufacturing Methodologies and Applications*, edited by W. D. Li, S. K. Ong, A. Y. C. Nee, and C. A. McMahon. Springer-Verlag, Berlin, Germany, 2007, pp. 211–243.

[109] Z. M. Qiu and Y. S. Wong. Dynamic workflow change in PDM systems. *Computers in Industry*, 58(5): 453–463, 2007.

[110] M. Saad and M. L. Maher. Shared understanding in computer-supported collaborative design. *Computer-Aided Design*, 28(3): 183–192, 1996.

[111] S. R. Fussell, R. E. Kraut, F. J. Lerch, W. L. Scherlis, M. W. McNally, and J. J. Cadiz. Coordination, overload and team performance: effects of team communication strategies. In: Proceedings of the ACM Conference on Computer-Supported Cooperative Work, Seattle, Washington, DC, November 14–18, 1998, pp. 275–284.

[112] G. Q. Huang, W. Y. Yee, and K. L. Mak. Development of a web-based system for engineering change management. *Robotics and Computer Integrated Manufacturing*, 17(3): 255–267, 2001.

[113] J. McGuire, D. Huokka, J. Weber, J. Tenenbaum, T. Gruber, and G. Olsen. SHADE: technology for knowledge-based collaborative engineering. *Journal of Concurrent Engineering: Applications and Research*, 1(3): 137–146, 1993.

[114] Y. Peng, T. Finin, Y. Labrou, B. Chu, J. Long, W. J. Tolone, and A. Boughannam. A multi-agent system for enterprise integration. In: Proceedings of PAAM'98, London, UK, 1998, pp. 213–229.

[115] H. Park, J. Tenenbaum, and R. Dove. Agile infrastructure for manufacturing systems: a vision for transforming the US manufacturing base. In: Proceedings of Defense Manufacturing Conference, 1993.

[116] P. M. D. Gray, S. M. Embury, K. Hui, and A. Prce. An agent-based system for handling distributed design constraints. In: Working Notes of the Agent-Based Manufacturing Workshop, Minneapolis, MN, 1998.

[117] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3): 38–49, 1992.

[118] W. Shen, F. Maturana, and D. H. Norrie. MetaMorph II: an agent-based architecture for distributed intelligent design and manufacturing. *Journal of Intelligent Manufacturing*, 11(3): 237–251, 1999.

[119] H. Park, M. Cutkosky, A. Conru, and S. H. Lee. An agent-based approach to concurrent cable harness design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 8(1), 45–61, 1994.

[120] C. Petrie, M. Cutkosky, T. Webster, A. Conru, and H. Park. Next-Link: an experiment in coordination of distributed agents. In: AID-94 Workshop on Conflict Resolution, Lausanne, Switzerland, 1994.

[121] C. Petrie. The Redux' server. In: Proceeding of International Conference on Intelligent and Cooperative Information Systems (ICICIS), Rotterdam, The Netherlands, 1993.

[122] Goldmann, S. Procura: a project management model of concurrent planning and design. In: Proceeding of WET ICE'96, Stanford, CA, 1996.

[123] S. E. Lander, S. M. Staley, and D. D. Corkill. Designing integrated engineering environment: blackboard-based integration of design and analysis tools. *Concurrent Engineering: Research and Applications*, 4(1): 59–72, 1996.

[124] N. E. Branki and A. Bridges. An architecture for cooperative agents and applications in design. In: *Advanced Technologies*, edited by M. R. Beheshi and K. Zreik. Elsevier Science Inc., New York, NY, USA, 1993, pp. 221–230.

[125] W. Shen and Q. Hao. A service oriented framework for blow molded automotive parts design and optimization. In: Proceedings of SAE 2004 Congress, Detroit, MI, SAE 2004-01-1244, 2004.

[126] R. Quadrel, R. Woodbury, S. Fenves, and S. Talukdar. Controlling asynchronous team design environment by simulated annealing. *Research in Engineering Design*, 5(2): 88–104, 1993.

[127] W. T. Tsai. Service-oriented system engineering: a new paradigm. In: Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE'05), 2005, pp. 3–6.

[128] W. D. Li and Z. Qiu. State-of-the-art technologies and methodologies for collaborative product development systems. *International Journal of Production Research*, 44(13): 2525–2559, 2006.

[129] V. Jagannathan, G. Almasi, and A. Suvaiala. Collaborative infrastructures using the WWW and CORBA-based environments. In: Proceedings of the IEEE Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE'96), 1996, pp. 292–297.

[130] G. Q. Huang and K. L. Mak. Web-based morphological charts for concept design in collaborative product development. *Journal of Intelligent Manufacturing*, 10: 267–278, 1999.

[131] A. Wallis, Z. Haag, and R. Foley. A multi-agent framework for distributed collaborative design. In: Proceedings of the IEEE Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE'98), 1998, pp. 282–287.

[132] Zdrahal, Z. and Domingue, J. The world wide design lab: an environment for distributed collaborative design. In: Proceedings of 1997 International Conference on Engineering Design, Tampere, Finland, 1997.

[133] N. H. M. Caldwell and P. A. Rodgers. WebCADET: facilitating distributed design support. In: Proceedings of IEE Colloquium on Web-Based Knowledge Servers, London, UK, 1998, pp. 9/1–9/4.

[134] G. Q. Huang, S. W. Lee, and K. L. Mak. Web-based product and process data modelling in concurrent 'design for X'. *Robotics and Computer-Integrated Manufacturing*, 15(1): 53–63, 1999.

[135] G. Q. Huang and K. L. Mak. Design for manufacture and assembly on the Internet. *Computer in Industry*, 38(1): 17–30, 1999.

[136] R. H. Allen, S. Nidamarthi, S. P. Regalla, and R. D. Sriram. Enhancing collaboration using an Internet integrated workbench. In: Proceedings of 1999 ASME Design Engineering Technical Conference, Las Vegas, NV, 1999.

[137] X. Q. Liu, S. Raorane, and M. C. Leu. A web-based intelligent collaborative system for engineering design. In: *Collaborative Product Design and Manufacturing Methodologies and Applications*, edited by W. D. Li, S. K. Ong, A. Y. C. Nee, and C. A. McMahon. Springer-Verlag, Berlin, Germany, 2007, pp. 37–58.

[138] F. Mervyn, A. Senthil Kumar, and A. Y. C. Nee. A 'plug-and-play' computing environment for collaborative product design and manufacturing across an extended enterprise. In: *Collaborative Product Design and Manufacturing Methodologies and Applications*, edited by W. D. Li, S. K. Ong, A. Y. C. Nee, and C. A. McMahon. Springer-Verlag, Berlin, Germany, 2007, pp. 71–92.

[139] G. Toye, M. Cutkosky, L. Leifer, J. Tenenbaum, and J. Glicksman. SHARE: a methodology and environment for collaborative product development. In: *Proceeding of 2nd Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE Computer Society Press, 1993, pp. 33–47.

[140] R. Fruchter, K. A. Reiner, G. Toye, and L. J. Leifer. Collaborative mechatronic system design. *Concurrent Engineering: Research and Applications*, 4(4): 401–412, 1996.

[141] M. J. Hague and A. Taleb-Bendiab. Tool for management of concurrent conceptual engineering design. *Concurrent Engineering: Research and Applications*, 6(2): 111–129, 1998.

[142] A. Varma, A. Dong, B. Chidambaram, A. Agogino, and W. Wood. Web-based tool for engineering design. In: Proceedings of AID'96 Workshop on Agents and Web-Based Design Environments, 1996.

[143] M. I. Campbell, J. Cagan, and K. Kotovsky. A-design: an agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design*, 11: 172–192, 1999.

[144] S. Balasubramanian, F. Maturana, and D. H. Norrie. Multi-agent planning and coordination for distributed concurrent engineering. *International Journal of Cooperative Information Systems*, 5(2–3): 153–179, 1996.

[145] M. Mahesh, S. K. Ong, and A. Y. C. Nee. A web-based framework for distributed and collaborative manufacturing of engineering parts. In: *Collaborative Product Design and Manufacturing Methodologies and Applications*, edited by W. D. Li, S. K. Ong, A. Y. C. Nee, and C. A. McMahon. Springer-Verlag, Berlin, Germany, 2007, pp. 141–154.

[146] Y. Wang, W. Shen, and H. Ghenniwa. WebBlow: a web/agent-based multidisciplinary design optimization environment. *Computers in Industry*, 52(1): 17–28, 2003.

[147] W. Shen and H. H. Ghenniwa. Multidisciplinary design optimization: a framework for technology integration. In: Proceedings of the First International Conference on Multidisciplinary Design Optimization, London, ON, 2001, pp. 22–28.

[148] W. Shen and H. H. Ghenniwa. A distributed multidisciplinary design optimization framework based on web and agents. In: Proceedings of the 2002 ASME DETC/CIE Conference, Montreal, Canada, September 29–October 2, 2002, DETC2002/CIE-34461, 2002.

[149] W. Shen. Web-based infrastructure for collaborative product design: an overview. In: Proceedings of 5th International Conference on CSCW in Design, Hong Kong, 2000, pp. 239–244.

[150] G. Q. Huang, J. S. K. Lau, and K. L. Mak. The impacts of sharing production information on supply chain dynamics: a review of the literature. *International Journal of Production Research*, 41(7): 1483–1517, 2003.

[151] A. Surana, S. Kumara, M. Greaves, and U. N. Raghavan. Supply-chain networks: a complex adaptive systems perspective. *International Journal of Production Research*, 43(2): 4235–4265, 2005.

[152] H. S. Jagdev and K. D. Thoben. Anatomy of enterprise collaboration. *Production Planning and Control*, 12(5): 437–451, 2001.

[153] L. M. Camarinha-Matos and H. Afsarmanesh. *Infrastructure for Virtual Enterprise*. Norwell, MA: Kluwer Academic Publisher, 1999.

[154] V. Marik and M. Pechoucek. Agent technology for virtual organization. In: Proceedings of PRO-VE'03, Lugano, Switzerland, 2003, pp. 243–252.

[155] N. Bakis, G. Aouad, and M. Kagioglou. Towards distributed product data sharing environments—progress so far and future challenges. *Automation in Construction*, 16(6): 586–595, 2007.

[156] W. Shen, Q. Hao, H. Mak, J. Neelamkavil, H. Xie, J. Dickinson, J. R. Thomas, A. Pardasani, and H. Xue. Systems integration and collaboration in architecture, engineering, construction and facilities management: a review. *Advanced Engineering Informatics*, 24(2): 196–207, 2010.

[157] R. Howard and B. C. Bjork. Building information modelling—experts' views on standardisation and industry deployment. *Advanced Engineering Informatics*, 22(2): 271–280, 2008.

[158] M. Sun, S. Senaratne, A. Fleming, I. Motowa, and M. L. Yeoh. A change management toolkit for construction projects. *Architectural Engineering and Design Management*, 2(4): 261–271, 2006.

[159] D. Tapscott and A. D. Williams. *Wikinomics: How Mass Collaboration Changes Everything*. New York, NY: Penguin Group, 2007.

[160] D. Richards. Collaborative knowledge engineering: socialising expert systems. In: Proceedings of CSCWD 2007, 2007, pp. 635–640.