Models for Internet-based Cooperative Engineering Design

Bogdan D. Czejdo
Department of Mathematics and Computer Science,
Loyola University
New Orleans, La. 70118, USA
czejdo@loyno.edu

Kenneth Messa
Department of Mathematics and Computer Science,
Loyola University
New Orleans, La. 70118, USA
messa@loyno.edu

Maciej Zakrzewicz
Department of Mathematics and Computer Science,
Loyola University
New Orleans, La. 70118, USA
mjzakrze@loyno.edu

Mikolaj Baszun
Institute of Microelectronics and Optoelectronics,
Warsaw University of Technology, ul. Koszykowa 75
Warsaw 00-662, Poland
baszun@imio.pw.edu.pl

Abstract

The paper discusses a cooperative engineering design process in which the design teams cooperate on design of engineering subsystems in order to optimize the performance of the entire system. We describe models for software systems supporting such engineering design. The software system consists of several coordination software modules that allow design teams to cooperate in the process of specifying or changing design decisions in any phase of the design process. Different interaction modes between the design teams are described. The implementation of such software system in the Internet environment is discussed.

Keywords: cooperative software, software modules synchronization

1. Introduction

The paper discusses a cooperative engineering design process in which the design teams cooperate on design of engineering subsystems (SCES) in order to optimize the performance of the entire system (CES). We assume that for each subsystem, the design should result in finding values of parameters fully describing the designed product [1, 2, 3]. These parameters are called *design parameters* (DP). There are numerous types of design parameters, such as material, geometrical, electrical and architectural. The design process is preceded by a detailed analysis of the physical phenomena resulting in specification of *constraints on the design parameters* (C_DP). The designed product has some behavioral properties. We refer to these properties as *output characteristics* (OC). Application, market, utility and other requirements for the engineering product result in some *constraints on the output characteristics* (C_OC). The goal of each subsystem design is to obtain design parameters resulting in "the best" output characteristics, which means those closest to the *ideal characteristics* (IC) while satisfying the constraints on the design parameters and constraints on the output characteristics.

Typically, each design team (shortly referred here as designer) uses some engineering computations module to assist in designing SCES. An overview of engineering computations software modules is given in the literature [3, 4]. For iterative design, usually they include some type of iterative Computation Control Module. There are still many challenges for engineering design software but many methods and results are already well established.

In order to efficiently coordinate the work of designer, some type of software coordination module is necessary. The area of software for cooperative tasks is in the phase of intensive research. In this paper we concentrate on coordination software modules for cooperative engineering design. Such modules allow many designers to cooperate in the process

of specifying or changing design decisions in any phase of the design process. Different interaction modes between the designers are taken into consideration.

There are several models that need to be developed while building software coordination module. First, the relationships between characteristics in subsystem designs need to be specified. We will define Characteristics Dependency Graph that describes them. Next, a graph needs to be constructed that shows the mode of interaction between designers for each group of related characteristics. We call this graph a Design Dependency Graph. We discuss two modes of interactions between designers. For each interaction mode the synchronization between different subsystems needs to be specified [5]. These can be done using the state diagram method as in [7, 8, 9, 11]. The state diagrams are then translated into software modules in the Internet environment.

2. Cooperative engineering design and its various modes

Complex engineering systems (CES) can consist of many subsystems (SCES). Each subsystem, in turn, can consist of many other subsystems. In cooperative engineering design, we assume that several designers are working on the design of separate subsystems. Initially, the main designer, in consultation with individual designers, provides a complete set of ideal characteristics and constraints on the output characteristics for each subsystem. Global constraints for the ideal characteristics are also specified.

Even though we assume that in a cooperative engineering design, several designers are working separately on the design of different subsystems, in many situations the real-time coordination of these designers can significantly improve the design of the complete system.

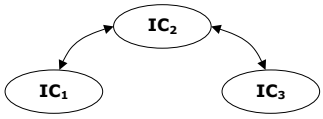

Figure 1. Characteristics Dependency Graph for a complex engineering system

The coordination can often result in dynamic modifications of some ideal characteristics (IC) and some constraints on output characteristics (C_OC) for many subsystems. Very often, a specific ideal characteristics (IC) is related with another ideal characteristics of another module in a sense that changes in the first would require changes in the second in order to satisfy the global requirements for the entire system. The same is true for constraints on output characteristics (C_OC). These relationships can be identified in the analysis phase of the engineering system. They can be described by Characteristics Dependency Graph as shown in Fig. 1. For example, IC1 and IC2 can be related in a sense that changes to IC1 will cause changes to IC2 and vice versa.

The Characteristics Dependency Graph can be converted into Initial Design Dependency Graph that shows design subsystems linked by characteristics pair relationships. Each characteristics pair relationship is created from an edge of Characteristics Dependency Graph. For example, the subsystem S1 and S2 are linked by relationship IC1-IC2 since any changes to IC1 in S1 have to be coordinated with changes to IC2 in S2 as shown in Fig. 2. Generally, each relationship between subsystems can be n-ary. Next, the Initial Design Dependency Graph is refined into Final Design Dependency Graph (or shortly Design Dependency Graph) that for each relationship shows the mode of interaction between designers.

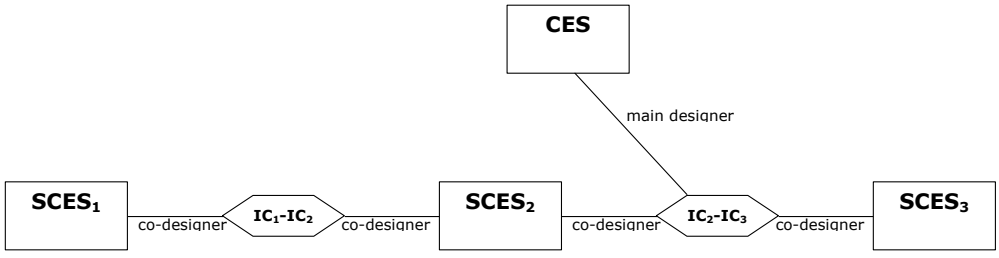

Figure 2. Design Dependency Graph for a complex engineering system

There are two modes of cooperation: **centralized** and **decentralized**. In the **centralized mode** all changes of requirements need to be approved by the main designer. The main designer or any other designer can be assigned of the role of a main designer for the specific pair of characteristics. This is reflected by inserting the label "Main-designer" as an appropriate relationship role. For example in Fig. 2 we assigned to the main designer the responsibility of coordination of design related with characteristics IC2 and IC3. In the **decentralized mode** all changes of requirements need negotiated by several designers. This is reflected by inserting the label "Co-designer" as an appropriate relationship role for more than one designer. For example in Fig. 2 we assigned the same responsibility to Designer 1 and Designer 2 related with characteristics IC1 and IC2.

The mode of cooperation between different subsystems for each relationship is selected based on potential propagation of changes and possible cycles. Generally, the **centralized mode** is chosen when the scope of potential propagation of changes is too large and possible cycles are detected in Characteristics Dependency Graph. Allowing for very extensive negotiations in this case would be inefficient if possible. Assigning one coordinator would allow to make all changes of requirements in much more orderly fashion. Usually, the **decentralized mode** for modifications of say, IC1-IC2, is chosen when extend of changes is limited to two subsystems and there is a well defined mapping between IC1 and IC2. If the mapping is complex and not straightforward that typically it is better to assign the coordinating role to the main designer.

3. Centralized Mode

The overall system architecture of the centralized mode of cooperative engineering design is shown in Fig. 3. The software system includes a Main Designer Coordination Module associated with the main designer. The Computation Control Module is responsible for engineering computations for the specific subsystem. Each Designer's Coordination Module interacts with the appropriate designer, Computation Control Module and other Coordination Modules. The sample interface for Coordination Module indicating the main designer's choices of actions is shown in Fig. 4a. The buttons are grouped into two categories. The first category, called Design, is for main designer's operations. **Suspend** is used to stop engineering computations in the appropriate Computation Control Module. **Change & Continue** is used to change the ideal characteristics or constraints on output characteristics. The second category, called Co-Design, is for the main designer to communicate his decisions on requests from co-designers. **OK Suspend** is used when the main designer accepts a co-designer's request to stop computations. **NO Suspend** is used when the main designer denies a co-designer's request to stop computations. Similarly, **OK Change& Continue** is used when the main designer accepts a co-designer's change in the characteristics, and **NO Change** is used when the main designer denies a co-designer's change in characteristics. The screen also includes the current state that informs the designer if automatic iterations are in progress, or if there is a request from co-designers. Additionally, data for design parameters and output characteristics are shown.

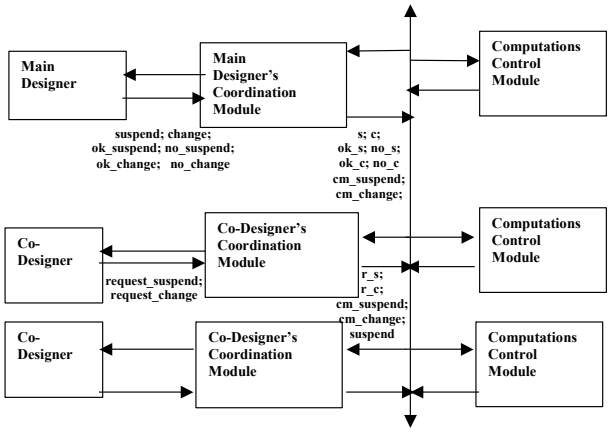

Figure 3. Centralized Mode of cooperative engineering design

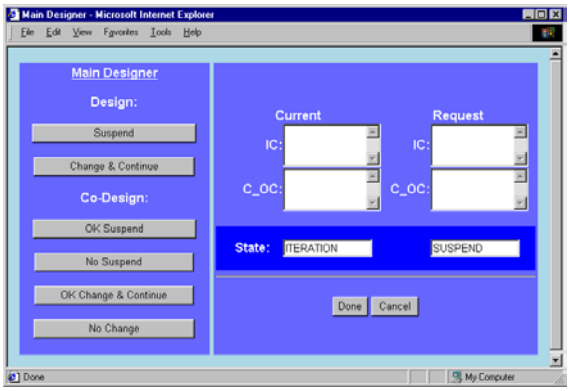

Figure 4a. Main Designer's Screen for Centralized Mode

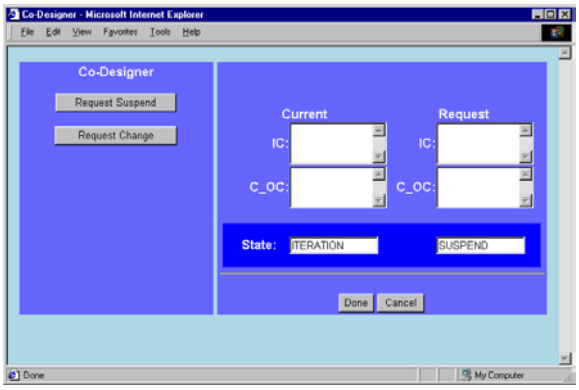

Figure 4b. Co-Designer's Screen for Centralized Mode

The co-designer's choices of actions are shown in a sample of a co-designer's screen in Fig. 4b. Since co-designers can only request action from the main designer, this interface has fewer choices. **Request Suspend** is used when the co-designer wants the main designer to stop iterations. **Request Change** is used when the co-designer wants the main designer to consider his change in parameters. **OK Suspend** is used when the co-designer wants suspend engineering computations. Similar to the main designer's screen, the co-designer is informed of values of output characteristics as well as the current state.

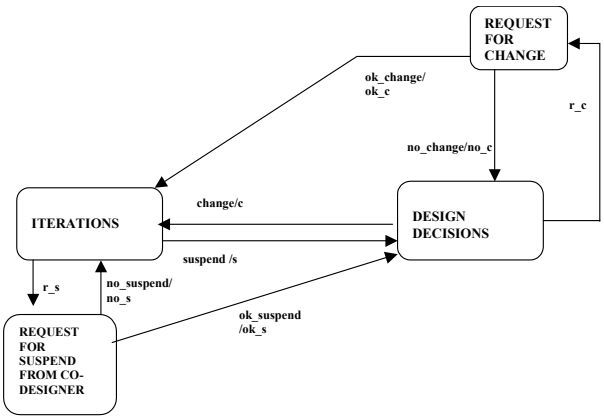

Figure 5a. Behavioral model for main designer's Coordinating Module

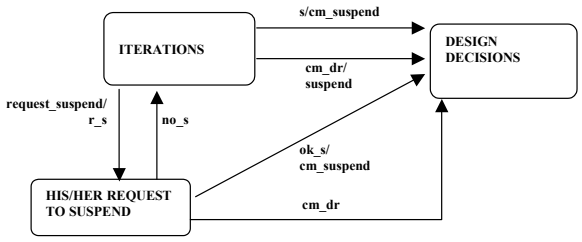

Figure 5b. Behavioral Model for Co-Designer's Coordination Module (Part A)

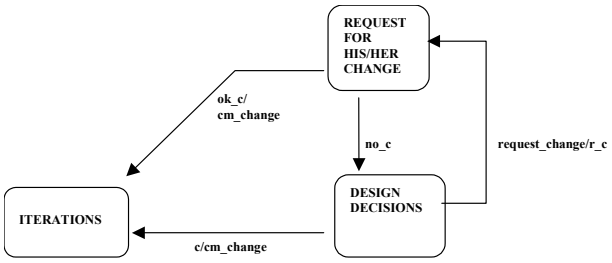

Figure 5c. Behavioral Model for Co-Designer's Coordination Module (Part B)

Description of behavior of all modules will be done using an extended state diagram [11]. Fig. 5a shows a simple object-behavior model for Main Designer's Coordination Module. Events can be generated by an interface such as one shown in Fig. 4 or by an action associated with the transition. We assume that the state diagram is executed a lock-step method with guaranteed delivery of the signals. That allows us to exclude from the model components responsible for lost or delayed signal such as acknowledgements, compensating transition etc.

Typically, the Main Designer's Coordinating Module will be in one of the two states: ITERATIONS or DESIGN DECISIONS. However, when the coordination is requested two additional transient states: REQUEST FOR SUSPEND FROM CO-DESIGNER and REQUEST FOR CHANGE FROM CO-DESIGNER are possible as shown in Fig. 5a

Co-Designer's Coordination Module can be in several states reflecting the current role of the co-designer and the phase of cooperation with the main designer as shown in Fig. 5b and 5c. Typically, the Co-Designer's Coordinating Module will be in one of the two states: ITERATIONS or DESIGN DECISIONS. However, when the coordination is requested two additional transient states: HIS/HER REQUEST FOR SUSPEND and HIS/HER REQUEST FOR CHANGE are possible as shown in Fig. 5b and 5c.

4. Decentralized Mode

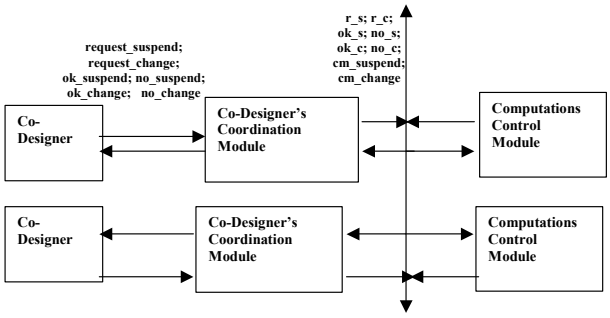

Figure 6. Decentralized Mode of cooperative engineering design

The overall system architecture of the decentralized mode of cooperative engineering design is shown in Fig. 6 The software system supporting this mode includes two software modules for each co-designer. These two modules include: Coordination Module and Computation Control Module. The Computation Control Module is responsible for engineering computations for the specific subsystem and is identical for both centralized and decentralized architectures. The sample interface for Co-designer Coordination Module indicating the co-designer choices of actions is shown in Fig. 7. The first two buttons are used to make a request for other co-designers. More precisely, **Request Suspend** is used to request stop engineering computations in the related Computation Control Modules. **Request Change** is used to change the ideal characteristics or constraints on output characteristics in the related subsystems. The second group of buttons is for the co-designer to communicate his decisions on requests from other co-designers. **OK Suspend** is used when the co-designer accepts another co-designer's request to stop computations. **NO Suspend** is used when the co-designer denies another co-designer's request to stop computations. Similarly, **OK Change** is used when a co-designer accepts another co-designer's change in parameters, and **NO Change** is used when a co-designer denies another co-designer's change in parameters. The screen also includes the current state that informs the designer if automatic iterations are in progress, or if there is a request from co-designers. Additionally, data for design parameters and output characteristics are shown.

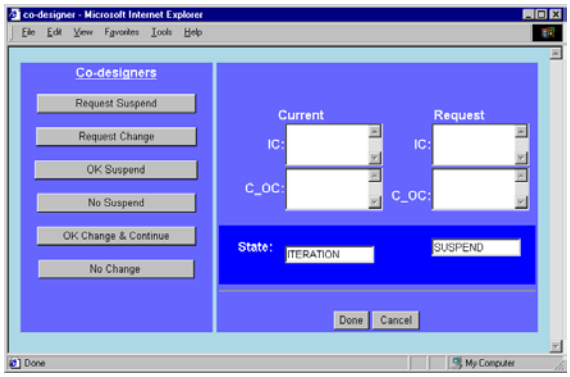

Figure 7. Designer's Screen for Decentralized Mode

If there is not complete agreement on either stopping engineering computations or changing parameters, the co-designer that introduced the request can modify the existing request or start a new request later.

The functionality of the Co-Designer's Coordination Module is a combination of functions performed by Coordination Modules in Centralized system. Therefore the diagram are some kind of superposition of diagrams in the Centralized system.

Co-Designer's Coordination Module can be in several states reflecting the phase of cooperation between co-designers. Typically, the Co-Designer's Coordinating Module will be in one of the two states: ITERATIONS or DESIGN DECISIONS. However, when the coordination is requested four additional transient states: HIS/HER REQUEST FOR SUSPEND, REQUEST FROM ANOTHER TO SUSPEND, HIS/HER REQUEST FOR CHANGE and REQUEST FROM ANOTHER TO CHANGE are possible as shown in Fig. 8b and 8c.

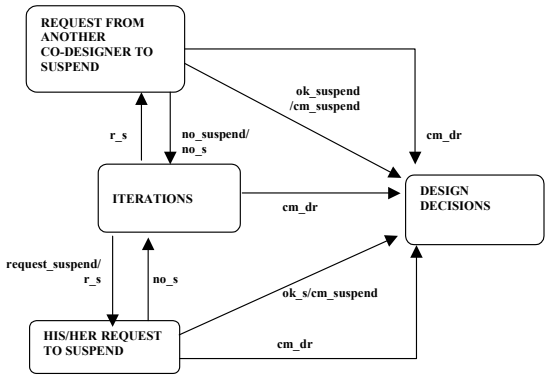

Figure 8a. Behavioral model for the decentralized mode (Part A)

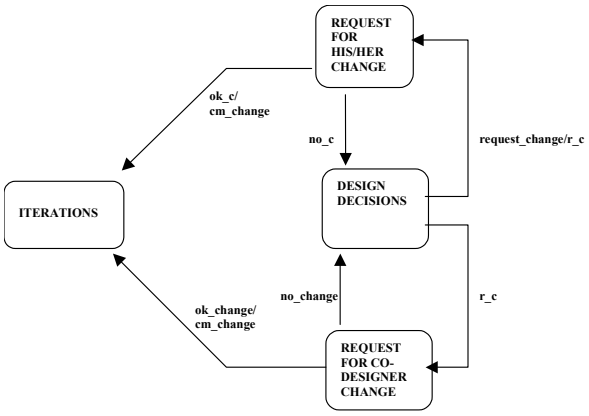

Figure 8b. Behavioral model for the decentralized mode (Part B)

5. Internet-based System Architecture

The cooperative engineering design system has been implemented as a collection of *Coordination Modules*. Each coordination module consists of two parts: the *Coordination Agent* (CA) and the *Coordination Gateway* (CG) (Fig. 9). Each coordination agent is responsible for: (1) cooperating with an external Computations Control Module and (2) communicating with other Coordination Agents.

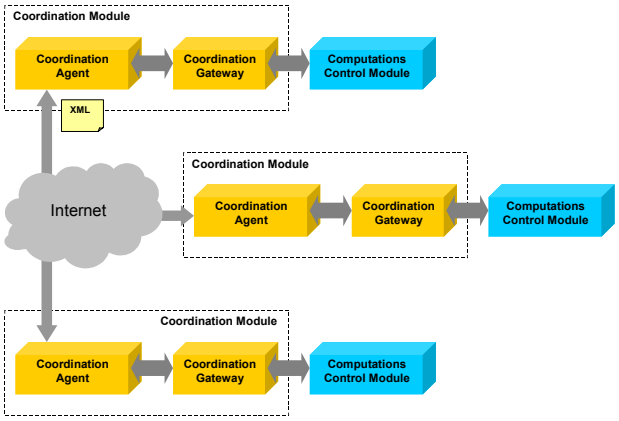

Figure 9. Internet-based System Architecture

Cooperation with external Computations Control Modules can be performed either manually or automatically. The manual cooperation involves a human operator, who reads Coordination Agent messages and issues the Computations

Control Module commands, using some separate user interface. The automatic cooperation requires using additional software modules, called *Coordination Gateways*, to directly access the programmatic interface of Computations Control Module. In this case the control commands can be send to the Computations Control Module by a Coordination Agent itself.

```
<xml>
  <request_change node_id="01"
                  priority="Main Designer"
                  coc_id="coc_182">
    <ic>
      <point x=3.1>1.897</point>
      <point x=3.2>1.922</point>
      <point x=3.3>1.998</point>
    </ic>
    <c_oc>
      <point x=3.1>
         <min>1.7<min><max>2.1</max>
      </point>
      <point x=3.2>
          <min>1.7<min><max>2.1</max>
      </point>
    </c_oc>
  </request_change>
</xml>
```

Figure. 10 Example XML message

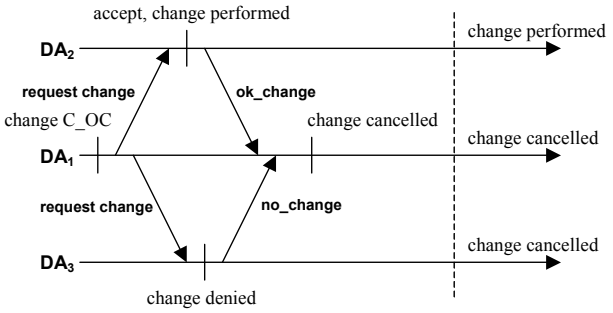

Figure. 11 Example of an incorrect state transition

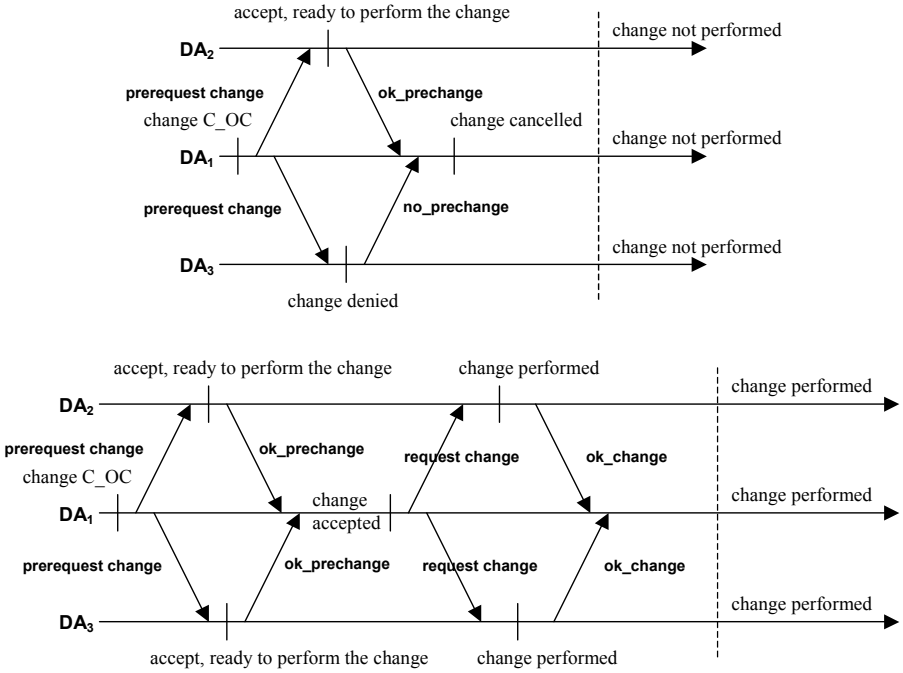

Figure 12. Two Phase Commit solution to the consistency problem

Inter-agent communication is implemented by means of *XML messages*, exchanged between the agents through the Internet. Example XML message is shown in Fig. 10. It consists of a message header and the message body. The message header contains information about the type of the request, the designer's id, the priority of the request, and the name of the characteristics. The message body contains information about the characteristics themselves.

The internet-based implementation introduces two additional problems: unknown and unlimited transmission time, and temporal inaccessibility of connections. In order to solve these problems, *confirmation messages* are introduced.

Upon receiving a command message, a design agent has to send a confirmation message to the sender. If the confirmation message does not reach the sender within a predefined time window, it is assumed that the recipient is currently not available. In this case, the command message is either ignored or stored in a buffer memory and retransmitted later. Since it may happen that a command message is successfully delivered but the confirmation message is lost, we additionally assign a unique id number to each message and therefore avoid command message duplication.

Additional problems may arise when the decentralized mode algorithm involves more than two design agents. Let us consider the following example of an incorrect system state transition given in Fig. 11. Assume three cooperating design agents, DA_1, DA_2, and DA_3. DA_1 wants to change the design constraint C_OC_1 but the other two agents must accept the change. So, DA_1 sends a **request_change** message to DA_2 and DA_3. DA_2 agrees to change C_OC_1, so it actually performs the change and responses with an **accepted_change** message. However, DA_3 denies the change, so it sends a **denied_change** message. After receiving the two messages, DA_1 realizes that there is no consensus, and therefore the change is cancelled on DA_1. Unfortunately, DA_2 has already changed C_OC_1, and the system state is inconsistent.

In order to solve this problem, a classical two-phase commit (2PC) approach can be employed. We add a new state called **prerequest_change** to the state diagrams. This state says that a designer agrees to perform the change when he or she receives the subsequent **request_change** message, but the actual change is not yet performed. The designer can also refuse the proposed change. The modified communication protocol is given in Fig. 12.

4. Summary

In this paper we introduced the architecture of a cooperative engineering system consisting of several design Coordination Modules and several iterative Computation Control Modules. Such a system allows many designers to cooperate in the process of specifying or changing design decisions in any phase of the design process. Different interaction modes between the designers were presented. A centralized mode of cooperative engineering was shown where a main designer and co-designers are identified. Additionally, a decentralized mode of cooperative engineering was shown where all designers were co-designers with equal control over operations.

We realize that in real life applications (e.g. mobile environments), the absolute consensus is not always required to change a C_OC. It should be generally possible that only *n* of *m* designers accept a change to force the change in the whole system. To implement such behavior, we allow some relaxation on decentralized mode rules. The minimum number of accepting nodes (MIN_ACCEPT property) can be specified for each C_OC, and then it is sufficient that only MIN_ACCEPT nodes accept the change.

References

[1] BASZUN M., CZEJDO B., "VSED - a Visual System for Engineering Design", *Proceedings of The First World Conference on Integrated Design and Process Technology*, Austin, Texas, USA, 6 – 9 Dec. 1995.

[2] BASZUN M., CZEJDO B., Miescicki J., "Modeling of Interactive Engineering Design", *Proceedings of ESDA'96*, Montpellier, France, 1 - 5 July 1996.

[3] BASZUN, M., MIESCICKI, J., AND CZEJDO, B., "Multilevel Modeling of Iterative Engineering Design of Remote Robot System," *Proceedings of The Second World Conference on Integrated Design and Process Technology*, Austin, 1996.

[4] BASZUN M., AND CZEJDO B. "Development of an Interactive Engineering Design Software", *Proceedings of The Third World Conference on Integrated Design & Process Technology*, Berlin, Germany, 1998.

[5] BEN-ARI, M., *Principals of Concurrent and Distributed Programming*, Prentice-Hall.

[6] CZEJDO B., DASZUK W. AND MIESCICKI J., "Concurrent Software Design Based on Constraints on State Diagrams", *Proceedings of The Third World Conference on Integrated Design & Process Technology*, Berlin, Germany, 1998.

[7] EMBLEY, D., KURTZ, B., WOODFIELD, S., "Object-Oriented Systems Analysis – a model driven approach", Prentice Hall, New Jersey, 1992.

[8] HAREL D., "Statecharts: A visual formalism for complex systems". *Science of Computer Programming,* 8: pp. 231 – 274, 1987.

[9] HAREL, D., "On visual formalisms", *Communications of the ACM*, 31(5), pp. 514-530, 1988.

[10] MIESCICKI J., BASZUN M., DASZCZUK W. B., CZEJDO B., "Verification of Concurrent Engineering Software Using CSM Models", *Proc.2nd World Conf. on Integrated Design and Process Technology*, Austin, Texas, USA, 1 - 4 Dec. 1996.

[11] RUMBOUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., LORENSEN, W., *Object-Oriented Modeling and Design*, Prentice Hall, New Jersey, 1991.