A Constraint Based Model of Coordination in Concurrent Design Projects

Lokesh Gupta, John F. Chionglo, Mark S. Fox
Enterprise Integration Laboratory, Dept. of Industrial Engineering, Univ. of Toronto, Toronto, Ontario, Canada, M5S 1A4
email: [gupta, chionglo, msf]@ie.utoronto.ca

Abstract

Communication and coordination play an important role in achieving concurrency in the design of large complex artifacts. It is also widely accepted that design is constraint oriented involving the recognition, formulation, and satisfaction of constraints. In this paper we have described how constraints can be used to achieve communication and coordination in large concurrent design projects. The system that we are implementing is oriented towards (but is not limited to) engineering domain, and supports multiple perspectives, notification mechanisms, a system management agent, and design knowledge management.

1 Introduction

Design of large complex artifacts involves efforts of many engineers with expertise from different disciplines. In order to structurally solve the big design problem, the system is divided into many sub-systems with well defined interfaces. Different teams of engineers work concurrently on these sub-systems, which later on merge together to build the complete system. Due to the interacting nature of the sub-systems, we need to make sure that whenever there are changes, there is proper communication and coordination of information. In case of large complex artifacts, no single engineer possesses the entire knowledge of the system. Therefore, it is extremely difficult and often impossible for an engineer to manually determine all the possible impacts of a given change. Hence, we need a fairly sophisticated mechanism of communication and coordination to be in place, which ensures that people get notified whenever they should have been notified, and that the efforts of various people on the project are coordinated well enough to absorb the impacts of the changes. This is the bottom-line to achieving high levels of concurrency on large design projects.

In case of fairly complex design projects, traditionally there have been two major approaches to achieving concurrency. These are (i) multi-disciplinary team *meetings*, and (ii) *distribution lists*. It has been found that both of these approaches fail to timely communicate and coordinate information (changes/decisions etc.), and hence, fail to achieve concurrency [10], [14].

The major problem with the distribution lists is that they are *static* in nature. Distribution lists are generally based on the names of the people and not their roles. However, who should be getting the information depends on who is *currently* playing the role which is relevant to that piece of information. Distribution lists would have served this purpose, had the relationship between the people, and their roles been static. However, the organization itself is quite *dynamic*. The roles and responsibilities of the people change quite often, and people get moved around. *It is because of this dynamic nature of organization, that the 'static' distribution lists fail to achieve concurrency.*

Design team meetings also fail to achieve high levels of concurrency on many counts. Following are some of them:

- In case of a complex change it is difficult to identify who all to invite in the meeting. One option is to call everybody in the meeting. However, studies on group dynamics say that teams function well only within a size of seven plus-minus two. Beyond that size, they start to become chaotic. Therefore, people are called selectively in the meeting. This immediately results in the loss of concurrency, because it is quite likely that a person who should have been there in the meeting has not been invited.
- Meetings are often wasteful of people's time, because only a small fraction of all the people are involved in a particular change [14]. For example, the software people are most likely, not to be bothered about the change in the stress of a particular mechanical equipment.
- Meetings often require physical co-location of the people. In case of the projects in which teams are located at geographically different places, physical

co-location is not only expensive, but also often impractical.

Because of the problems described above, we need another approach which can help in achieving high levels of concurrency in the design of large complex artifacts. In this paper we describe a constraint-based system which attempts to achieve high levels of concurrency by timely communication and coordination of information. We believe that *communication and coordination should be based on the impacts of the changes and the responsibilities of the people in the design project*. The impact of a change determines what to communicate and who to communicate it to. In order to do so, we also need a very broad scope of changes. Our definition of change, does not only include, for example, changes to the numeric values of the design parameters, but also includes version releases, work in progress, organizational changes etc. And, the means to identify who are impacted by a change is through the propagation of change through a network of constraints. Having identified who are impacted, the last step in achieving concurrency is to notify these people, and coordinate their efforts to absorb the impacts of the change.

This constraint-based system of communication and coordination, is a part of a larger project called Knowledge Aided Design (KAD) [6]. In Section 2, we describe the system architecture of KAD. In Section 3, we describe our model of concurrency. In section 4 we describe a classification of impacts. Section 5 describes our approach to implement the model of concurrency. In the sections that follow thereafter, we describe the details of our implementation. Section 6 describes the Enterprise Object Model (EOM). Section 7 describes our approach to finding out who are impacted by a given change. In section 8, we describe the "Change-Symptom-Action" approach, and its implementation using the constraint manager (CM), and the systems management agent (SMA). We conclude with a brief summary of our work in Section 9.

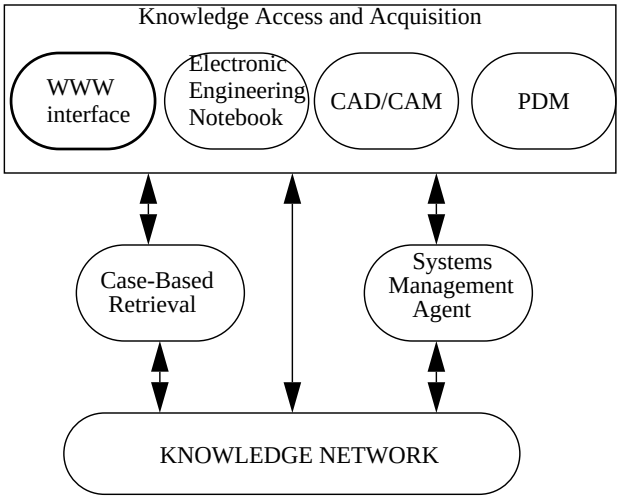
2 KAD Architecture

At University of Toronto's Enterprise Integration Laboratory, we are constructing a Knowledge Aided Design (KAD) system for supporting concurrent engineering design and enhancing the degree of awareness, cooperation, and coordination among engineering team members. The main objectives of the KAD system are to:

- Provide a shared representation that stores, integrates and manages various types of design knowledge. The representation should be able to model and represent information such as design rationale, requirements, versions, product structure, organization etc., and provide the ability to reason about them. The environment should not only provide each engineer with a private working space where the engineer can explore his design at his will, but also protocols to integrate the works of different engineers into the shared representation.
- Ease the access (to find the existing information) to and acquisition (to get the information out of engineers head, back-of-paper, notes etc.) of information and knowledge from the representation and to allow reuse of existing design knowledge.
- Integrate existing tools like CAD/CAM, PDM etc. with the KAD system in a seamless manner. The KAD system should provide services for uploading, and downloading of information to and from these tools.
- Manage the Systems Engineering process by providing adequate communication and coordination capabilities, in order to improve the productivity and quality of the design process.

The KAD system architecture is shown in Figure 1.

Figure 1 KAD Architecture



The Knowledge Network (KN) provides the capability for storing design knowledge and services for the management of shared design. The Case-Based Retrieval (CBR) module provides the capability of re-using the existing design knowledge. The knowledge access and acquisition module provides a World Wide Web (WWW) and an Electronic Engineering Notebook (EEN) interface for knowledge access and acquisition. It also integrates the other tools to the KAD system. The Systems Engineering Management Agent (SMA) is a rule-based system which manages the communication and coordination of information.

In the context of this paper, only SMA, and the enterprise object model, and constraint management layers of KN are of relevance (cf. Figure 3). As such we have described only these components of KAD architecture

here. The other portions of the system architecture are reported elsewhere [1], [8].

3 Model of Concurrency

As a part of our research we have investigated what needs to be done in order to achieve a high levels of concurrency in large design projects. Based on a set of interviews[1] [10] that we have conducted, we found that:

- Timely communication of accurate information plays a very important role in achieving concurrency
- People need to share their work in progress for achieving concurrency
- Changes are the prime driver for the requirement of concurrency. As such the impacts of the changes need to be determined accurately and the actions of various experts need to be well coordinated in order to absorb those impacts.

One of the biggest hurdles in the achievement of concurrency is that people often do not know, who to inform when they are making a change. This is especially true of complex designs. Most of the current mechanisms of communication and coordination are manual in nature and fail on the ground that no single person understands the system well enough to know about all the possible impacts of the change [14]. Conflicts are usually detected later in the product life-cycle when it is very costly to resolve them.

Our model of concurrency is, therefore, based on the above listed points, and it states that:

- ***Ensure that people share their work in progress and that everyone is working off the same piece of data.***
- ***Whenever, there are changes, find out the impact of the change.***
- ***Inform all the people who can be affected by the change, and coordinate their efforts to absorb the impacts of the change***.

The model describes what needs to be done in order to achieve concurrency and not how to achieve it. There could be many possible implementations of this model of concurrency. As a part of this work, we have described one such implementation which takes a very broad view of changes, provides a common language to people to express their design decision, and initiates necessary communication and coordination activities to achieve high levels of concurrency.

1. A total of 13 interviews were conducted across 4 different organizations.

4 Impact Classification

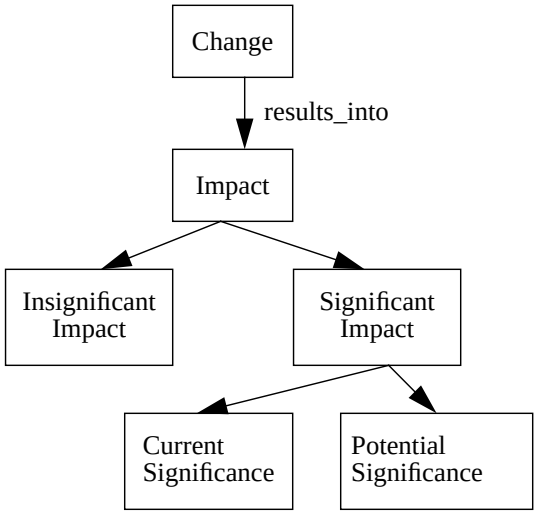
We propose that communication and coordination should be based on:

- the *impact* of the changes, and
- the *responsibilities* of people across the project

In this section we give a classification of impacts, and some examples of impact-based communication. Responsibility-based communication is described in section 8.2.

A change may not always result in a significant impact. Therefore, we need to distinguish between various types of impacts that a change may result into. Our classification of impacts is shown in Figure 2.

Figure 2 Impact Classification



We give below some of the examples[2] which illustrate the above classification:

- Insignificant Impact: For example, assigning an additional role, R, to an engineer, E. This change may not impact anyone else's work, and just needs a simple notification to all the members of the project that engineer E is now playing the role R. Another example could be the release of a new version of a drawing. The drawing may not be in conflict with any existing designs, but still we need to inform people that now there is a new version available. The new version of the drawing may have some future impact.

2. We would like to make a note here that we have just cited some examples. We are not attempting to say that the examples we have given above fall into the corresponding impact category. On the contrary, we believe that it is not possible to determine the impact of a change in advance. Impact of a change is not only a function of the change itself, but also a function of the current state of the world. Therefore, the impact needs to be determined dynamically.

In that case, it would be classified as an impact of potential significance.

- Impacts of Current Significance: The most common ones are conflicts. In such cases, we need to know what is the impact, who are impacted, and then take necessary steps to resolve the conflict. Much work has been done in the area of conflict detection and resolution [13]. Another example could be the Work-In-Progress (WIP) of an engineer. The WIP may be related to the works of other people in the project, and hence, needs to be communicated to all of them.
- Impacts of Potential Significance: Some of the changes, may not be causing any impacts of current significance, but may eventually cause some problems. For example, it may be the case that weight of the artifact being designed is an important parameter to be watched, and it has been observed that for the last three design reviews the weight has been consistently increasing by 15%. At present, this change may not be causing any conflicts with the overall weight requirement, but it may be an indicator of a possible future violation of the requirement.

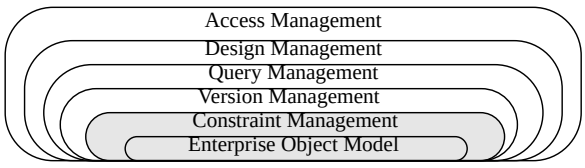
5 An Implementation of the Model of Concurrency

In order to implement the model of concurrency we propose to have:

- a shared object model which integrates product, organization, and processes. Within that representation we have means to represent requirements, constraints, parameters, roles, responsibilities, etc.
- a systems management agent which handles the communication and coordination mechanism based on the messages it receives from the shared object model.

Our shared object model, called Knowledge Network (KN), is a layered architecture (Figure 3).

Figure 3 Knowledge Network



The Knowledge Network serves the needs of the entire KAD project. In the context of this paper, the layers that are of relevance to us are the enterprise object model (EOM) layer, and the constraint management layer. EOM provides (i) a common language (in the form of its objects) for people to work, and (ii) a medium to make sure that everybody is working off the same piece of data, and that people are able to share their work in progress. This takes care of the first portion of the implementation of our model.

Changes are made by various people to the objects of EOM. We have adopted a fairly broad scope of changes, which includes, work in progress, version releases, critical project parameters, organizational changes etc. Based on the changes made to the EOM layer, the constraint manager (CM) finds out the impact of the change, and sends messages to SMA. Based on the work that people are currently doing, and the responsibilities of the people on the project, SMA finds out who needs to be notified. Thus, the constraint management layer along with SMA implements the impact- and responsibility-based communication and coordination aspect of our model.

The remainder of this paper describes each one of these concepts in detail.

6 Enterprise Object Model

The core of the Knowledge Network is an enterprise object model developed within the context of TOVE (TOronto Virtual Enterprise) project, [7]. The object model provides a representation of data/knowledge spanning product, process and organization. In addition, it includes rich representation of activity, time, causality, resources, cost, quality and more. Object class library for engineering design provided by the enterprise object model includes:

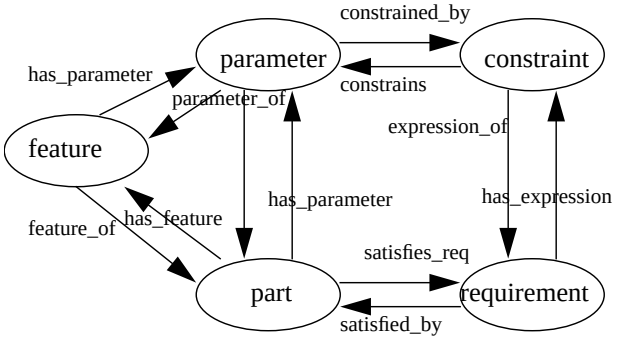
- Component structure, parts, features, parameters,
- Requirements, constraints, dependency, source,
- Versions,
- Decisions, rationale, alternatives

Component structure is represented by *parts* connected with the relationship “has_component”. Features are used to describe geometrical and functional characteristics associated with a part. Both part and feature can have parameters that define their properties such as weight, color, diameter, material, surface finish, etc. Requirements specify the properties (functional, structural, physical, etc.) of the artifact being designed. Constraints form the leaves of the requirement decomposition tree, and embody physical laws, equations etc.[1] Thus, there exists a network of information. ***It is this network that we surf to find the communication and coordination links***.

The relationships between parts, features, parameters, requirements and constraints are shown in Figure 4.

1. This representation is based on the concepts described in [16].

Figure 4 Representing Design Knowledge

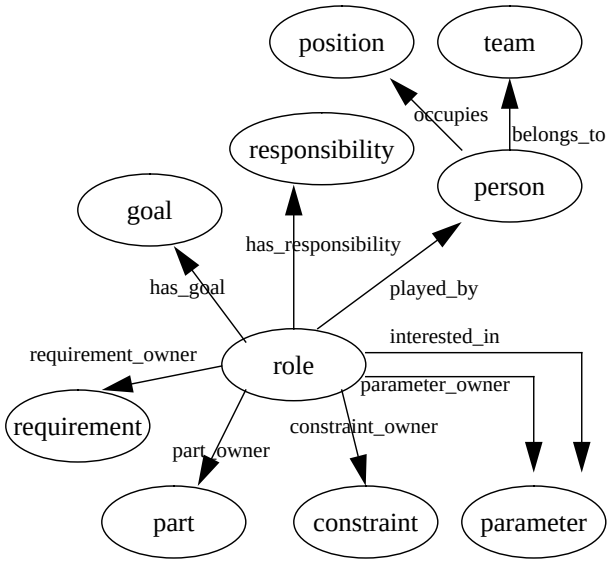


The enterprise object model includes the following representation (ontology) about project/organization:

- Teams, resources
- Responsibility, authority, role, position
- Interactions, dependencies
- Activities, goals, milestones.

Constraints, parameters, parts etc. are owned by different organization roles. Each role is played by a person. A person occupies a position, and belongs to one or more teams. Relationships between the organization and product ontology is shown below in Figure 5.

Figure 5 Organization Ontology and Its Relationship with Product Ontology[1]



1. We have not shown all the possible links that exist in our Knowledge Network. Only some of the links have been shown to give the reader an idea of the integrated representation.

7 How to determine who will be affected?

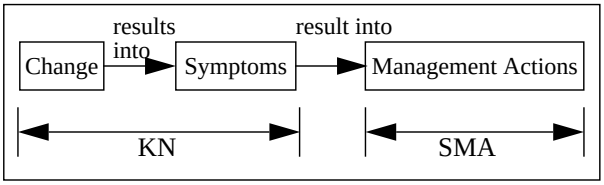
In case of a concurrent engineering environment, the design is being simultaneously considered from various life-cycle perspectives. Each one of these perspectives puts forward some criteria that the design must satisfy. Constraints provide a common language for these experts to talk to each other. Not only do these perspectives impose constraints on the design, they also try to push forward their concerns ahead of that of the others [3]. The constraints imposed by one perspective may conflict with the constraints imposed by another perspective. Design can therefore be modelled as a network of constraints, and *we can make use of the constraints to find out the people who are impacted by a given change*.

Most of the previous work in the area of constraints has been in the area of design "solving" [2], [4], [9], [17], [18]. Recently researchers have started to look into the area of supporting concurrent engineering using constraints [5]. The novel aspect of this work is that it goes one step further, in the sense that it ***not only supports*** concurrent engineering (by allowing the user to have multiple perspectives, constraint propagation etc.), ***but also provides mechanisms to achieve high levels of concurrency*** by timely communication and coordination of information (who to inform, what to inform, when to inform etc.).

8 Change-Symptom-Action Approach

We adopt a "Change-Symptom-Action" approach to communicate between the constraint manager (CM) and SMA. CM checks for some of the symptoms that arise as a result of a change, and informs SMA. Based on the type of the symptom SMA takes necessary actions to manage the impacts of the change. The "Change-Symptom-Action" approach is shown in Figure 6.

Figure 6 Change-Symptom-Action Approach



8.1 Constraint Manager

In the current version of our implementation, the constraint manager works only on numeric parameters. We list below some of the key aspects of the constraint manager:

8.1.1 Interval Propagation

All of our parameters have interval values allowing the design to proceed with incomplete information. Often, designers do not know the exact value that a parameter will get, until the detailed design phase [12], [15]. However, they have some idea of the limits within which the value will fall. Thus, we need to have interval domains of the parameters, and the capability of doing constraint propagation on interval variables.

8.1.2 Modes of operation

CM operates in more than one mode. The user may like CM to work as a "consultant", in which case it performs an impact analysis and produces a report for the user. Or, the user may like CM to act as a decision maker, in which case it will implement the effects of the change and reflect it in the new state of the design. In the current version, the user sets a flag to "analysis" or "implement" to switch between these two modes. However, in the future versions when the access management layer of the Knowledge Network is fully operational, we plan to derive the authority of switching between these flags, based on the roles and responsibilities of the user. For example, only the person responsible for the design will be able to switch to the "implement" mode of operation of the constraint manager.

8.1.3 Analyze a single change or multiple changes

The user can analyze one change at a time, or build a stack of changes over a period of time and analyze them all together. We also store the state of the design before a change was made. Using this mechanism a user can always revert back to the previous state, i.e the state, before the change was made.

8.1.4 Impact Analysis at Different Levels of Design

The user can study the impacts of a change at three different levels:

- Frame (or Parameter or Object) Level: Start from the actual parameter frame that has been changed and do constraint propagation. This capability is provided, because an engineer working on a component, and thinking of making a change may like to study the affect of that specific change which he/she is making to the parameter frame.
- Design Set Level: Design set is like a local space for the engineer to work on the designs, and is described in detail in [8]. In this level the constraint manager considers all the parameters and constraints in the design set and does constraint propagation. Analysis at the design set level is provided, because an engineer may like to study the consistency of his design within the design set, before submitting the design and making it available it to the world.
- Design Level: In this case CM studies the whole design to find out what constraints are being violated and how will the simultaneous consideration of all the constraints will effect the parameter values. The design level is provided to make sure that all the designs within the systems are consistent. In other words, a mechanical design may be consistent within itself, but may be in conflict with the electrical design. These inconsistencies can be revealed by running the constraint manager at the design level.

The constraint manager is built using ECRC's (European Computer-Industry Research Centre) constraint logic programming system ECLiPSe [19] and the CLP(Q,R) constraint solver [11], on a Sun SPARC workstation running SunOS version 4.1.3.

8.2 Systems Management Agent (SMA)

SMA embodies the model of coordination in the form of rules regarding how to manage a project in order to achieve high levels of concurrency. This is an independent process which communicates with the knowledge network through a set of messages.

The current implementation of SMA notifies engineers, and creates activities in their to-do lists whenever there are impacts on the parameters and constraints that they are responsible for. We illustrate this with an example. In case of space systems designs, weight is an important parameter that needs to be observed very carefully. As a result of some changes, if the domain of a weight parameter shrinks, then SMA immediately informs the weight engineer, and advises him to make sure that he can still proceed with the new limits on the weight parameter. SMA also informs the weight engineer about the origin of the change, and who has initiated the change. If the weight engineer finds out that these new limits are too restrictive for the design of his part, he can discuss the matter immediately with the initiator of the change. If the new limits are okay with the weight engineer, then also SMA has performed its duty of timely communicating the relevant information to the weight engineer, and thereby enhancing the level of concurrency in the project.

SMA also contains rules regarding responsibility-based communication. For example, the project leader may not be directly involved in the design of a particular component, but since he has the responsibility of the overall project coordination we send notifications to the project leader whenever there are "significant" changes to the design. Significance of a change is determined on the basis of the project management rules put into the SMA.

Another example is a scenario in which the cost of the project is showing a very sharp increase. In this case, we inform the senior management people responsible for allocating the cost budgets to the projects. This provides, for example, a ***risk and cost management capability*** in SMA.

SMA is implemented in CLIPS (C Language Integrated Production System) Version 5.1. CLIPS is a rule-based language which can be used building sophisticated expert systems.

9 Summary

We have described a model of concurrency, and an implementation of the model. Achieving high levels of concurrency requires that people share their work, and that impacts of changes are informed to all the people who can be affected. Our implementation is based on a shared representation called the Knowledge Network (KN), and a systems management agent (SMA). KN provides an environment for people to share their work, and work on same piece of data. SMA embodies the model of concurrency in the forms of rules that can be very general or very specific to the way a particular organization manages its projects. Based on its rule-base, SMA carries out impact- and responsibility-based coordination. Thus, SMA along with KN provides a very flexible, and effective mechanism for achieving high levels of concurrency.

References

[1] Bilgic, T., and Fox, M. S. ; (1996) ; “Constraint-Based Retrieval of Engineering Design Cases: Context as Constraints” ; Artificial Intelligence in Design’96, Stanford University, June 24--27, 1996, (to appear in S. Gero and F. Sudweeks (eds) AI in Design’96, Kluwer Academic Publishers)

[2] Borning, A., and Duisberg, R. ; (1986) ; “Constraint-Based Tools for Building User Interfaces” ; ACM Transactions on Graphics, Vol. 5, No. 4, October 1986 ; pp. 345-374

[3] Bowen J., and Bahler, D. ; (1994) ; “An Axiomatic Approach That Supports Negotiated Resolution of Design Conflicts in Concurrent Engineering” ; In: Artificial Intelligence in Design’ 94, Edited By: J. S. Gero, and F. Sudweeks ; pp. 363-379

[4] Ervin, S. M., and Gross, M. D. ; (1987) ; “ROADLAB - A Constraint Based Laboratory for Road Design” ; In: Knowledge Based Expert Systems in Engineering: Planning and Design, Edited By: D. Sriram, and R. A. Adey ; pp.167-184

[5] Fohn S. M., Greef A. R., Young R. E., and O’Grady P. J. ; (1994) ; “A constraint system shell to support concurrent engineering approaches to design” ; In: Artificial Intelligence in Engineering, 9:1-17, 1994.

[6] Fox, M., S., Bilgic, T., Lin, J., Gupta, L., Jacek, G., Chionglo, J., F., and Leizero, W. ; (1996) ; “Knowledge Aided Design” ; Working Paper, Enterprise Integration Laboratory, Department of Industrial Engineering, University of Toronto, Toronto, Canada M5S1A4

[7] Fox, M., Chionglo, J.F., and Fadel, F., G., ; (1993) ; “A Common Sense Model of the Enterprise” ; In: Proceedings of the 2nd Industrial Engineering Research Conference, Norcross GA, Institute for Industrial Engineers ; pp.425-429

[8] Fox, M. S, Lin, J., and Gupta L. ; (1996) ; “Knowledge Network: An Information Repository with Services for Managing Concurrent Engineering Design” ; Technical Report, Enterprise Integration Laboratory, Department of Industrial Engineering, University of Toronto, Toronto, Canada M5S1A4

[9] Gossard, D. C., and Serrano, D. ; (1985) ; “MATHPAK: An Interactive Preliminary Design Package” ; In: Computers In Engineering, Proceedings of the 1985 ASME International Computers In Engineering Conference and Exhibition, Volume 3, August 4-8, 1985, Boston, Massachusetts ; pp. 221-226

[10] Gupta, L. ; (1996) ; “Thesis Interviews - Coordination and Concurrency” ; Technical Report, Enterprise Integration Laboratory, Department of Industrial Engineering, University of Toronto, Toronto, Canada M5S1A4

[11] Holzbaur, C. ; (1995) ; “OFAI clp(q,r) Manual” ; Technical Report TR-95-09, Austrian Research Institute for Artificial Intelligence, Vienna. Edition 1.3.3.

[12] Kim, K., Cormier, D., O’Grady, P., J., Young, R. E. ; (1994) ; “A System For Design and Concurrent Engineering Under Imprecision” ; Technical Report, TR-095, Group For Intelligent Systems in Design and Manufacturing, Department of Industrial Engineering, North Carolina State University, Raleigh, North Carolina, 27695-7906, USA

[13] Klein, M. ; (1990) ; “Conflict Resolution in Cooperative Design” ; In: The International Journal For Artificial Intelligence in Engineering, Vol.4, No.4 ; pp.168-180

[14] Klein, M. ; (1994) ; “Computer-supported Conflict Management In Concurrent engineering: Introduction to Special Issue” ; In: Concurrent Engineering - Research and Applications, Volume 2, September 1994 ; pp. 145-147

[15] Krishnan, V., Navinchandra, D., Rane, P., and Rinderle, J., R. ; (1990) ; “Constraint Reasoning and Planning in Concurrent Design” ; Technical Report, CMU-RI-TR-90-03, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

[16] Navinchandra, D., Fox, M., S., and Gardner, E., S. ; “Constraint Management in Design Fusion” ; To appear in: Concurrent Engineering, Edited by: P. Gu, and A. Kuisak

[17] Sriram, D., and Maher, M. L. ; (1986) ; “The Representation and Use of Constraints in Structural Design” ; In: Applications of Artificial Intelligence in Engineering Problems, April 1986, Vol. I, 1st International Conference, Southampton University, UK, Springer-Verlag ; pp. 355-368

[18] Sussman, G. J., and Steele, G. L. Jr. ; (1980) ; “CONSTRAINTS - A Language for Expressing Almost-Hierarchical Descriptions” ; In: Artificial Intelligence 14 (1980) ; pp.1-39

[19] Wallace, M., and Veron, A. ; (1993) ; “Two problems -- two solutions: One system -- ECLiPSe” ; In: Proceedings IEE Colloquium on Advanced Software Technologies for Scheduling, London.