COOPERATIVE MODELING AND DESIGN HISTORY TRACKING USING DESIGN TRACKING MATRIX

A Thesis

by

JONGHYUN KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2009

Major Subject: Industrial Engineering

COOPERATIVE MODELING AND DESIGN HISTORY TRACKING USING DESIGN TRACKING MATRIX

A Thesis

by

JONGHYUN KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2009

Major Subject: Industrial Engineering

ABSTRACT

Cooperative Modeling and Design History Tracking

Using Design Tracking Matrix. (August 2009)

Jonghyun Kim, B.S., Korea Military Academy

Chair of Advisory Committee: Dr. Amarnath Banerjee

This thesis suggests a new framework for cooperative modeling which supports concurrency design protocol with a design history tracking function. The proposed framework allows designers to work together while eliminating design conflicts and redundancies, and preventing infeasible designs. This framework provides methods to track optimal design path and redundant design history in the overall design process. This cooperative modeling architecture consists of a modeling server and voxel-based multi-client design tool. Design change among server and multiple clients are executed using the proposed concurrency design protocol. The design steps are tracked and analyzed using Design Tracking Graph and Design Tracking Matrix (DTM), which provide a design data exchange algorithm allowing seamless integration of design modifications between participating designers. This framework can be used for effective cooperative modeling, and helps identify and eliminate conflicts and minimize delay. The proposed algorithm supports effective cooperative design functions. First, it provides a method to obtain the optimal design path which can be stored in a design library and utilized in the future design. Second, it helps capture modeling pattern which can be used for analyzing designer’s performance. Finally, obtained redundancies can be used to evaluate designer’s design efficiency.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Amarnath Banerjee for his kind guidance, encouragement and advice during this research. I wish to extend my thanks to Dr. Lewis Ntaimo and Dr. Salih Yurttas for serving on my graduate committee. I am also grateful to the faculty members of the Department of Industrial Engineering at Texas A&M University. Thanks also to Mr. Hyun Soo Lee for his suggestion which made this thesis much better.

Finally, I would like to thank my dear wife, Mijae Kim, who kept me focused on finishing this research and my lovely son, Justin (Gio).

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction

A number of cooperative modeling technologies exist in industrial product design environments providing real-time sharing and modification methods. Some of their advantages are to shorten overall design time using instantaneous consensus and coordination among multidisciplinary designers. Also, these software provide methodologies for designers/engineers to share design resources and evaluate product design feasibility. From this point of view, Computer Supported Cooperative Work in Design (CSCWD) has played an important role in product design. In addition, CSCWD and Concurrent Engineering (CE) tools focus on providing environments which allow users to work together on a single task [1-3]. Although, these existing frameworks are good for cooperative modeling, there are some limitations. The existing frameworks/systems do not support sufficient history tracking ability. In cooperative modeling, design history tracking is an essential function for checking redundant design, backtracking, and acquiring desired design in minimal time. However, many Product Lifecycle Management (PLM) software such as PTC's *Windchil/PDMlink*, UGS's *Teamcenter*, IBM's *ENOVIA* just provide a simple tracking document change history functionality. Another limitation is that these products fail to capture each designer's modeling pattern for evaluating designer's performance and utilizing it subsequently. A designer's modeling pattern can be used for the next modeling task. If we know a

---

This thesis follows the style of *IEEE Transactions on Automatic Control.*

designer’s modeling pattern, it can be stored in a design library and utilized in similar product design. It is difficult to know the preference of each designer, as well as the differences between designers’ patterns.

Cooperative modeling with many designers is one of the good methods for minimizing design time and designing sophisticated products. In this context, the objective of this research is to propose (i) a cooperative modeling framework based on voxel-based cooperative modeling, (ii) a method to manage data resources and tracking history using a proposed Design Tracking Matrix (DTM), and (iii) a method to analyze the design using a proposed Numerical DTM.

1.2 Thesis Outline

The remainder of this thesis is as follows. Section 2 introduces the background related to this research. This section includes design method, concurrency management protocol, Petri net and design structure matrix. In section 3, the cooperative modeling environment is presented. This section includes voxel-based cooperative system architecture and developed concurrency management protocol using Petri net described in section 2. In section 4, the algorithm for design history tracking is proposed which includes the method to obtain optimal design path using design tracking matrix. In section 5, the algorithm for design analysis is discussed and it includes the method to represent the design tracking matrix in a numerical representation and its utilization. The conclusion is discussed in section 6, which summarizes this thesis and highlights the contributions of this thesis.

2. RESEARCH BACKGROUND

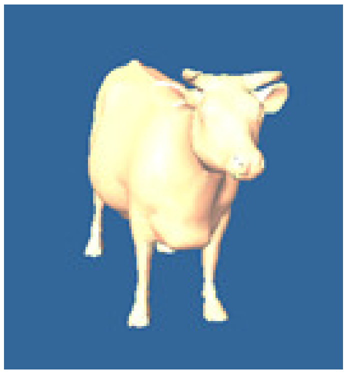
2.1 Voxel-Based Design

The collaborative design framework and design tracking algorithm is based on voxel-based design and representation. The term, voxel is derived from **vo**lumetric pi**xel**. While common image pixel has 2 dimensions (x and y), a voxel has 3 dimensions (x, y, and z). In general, one voxel is represented as a cube of unit length. However, a voxel model can have variations such as a voxel model with different dimension sizes or subdivided voxel model similar to an octree representation [4]. Figure 1 shows an original cow shape and its voxel representation.

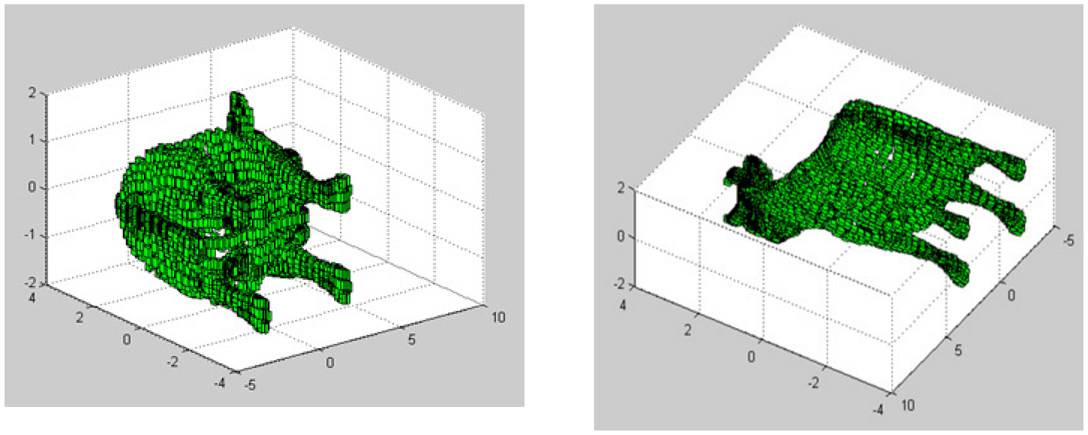
Since voxel model is a simple representation metholodgy in modeling and design, it has been widely used for approximating complex shapes, capturing shapes quickly or creating prototypes. Compared to B-rep, CSG and other non-manifold representations, it has some advantages and disadvantages [5, 6]. Table 1 summarizes some of the characteristics of voxel-based design.

Table 1: Charateristics of Voxel-Based Design

	Characteristics
Voxel-based design	1. Easy boolean operation 2. Excellent local editing ability 3. Insensitivity to object complexity and topology 4. Useful for feature recognition



(a) Original 3D shape



(b) Voxel representation of the cow model

Figure 1: Original 3D Shape of a Cow Model and Its Voxel Representation

2.2 Concurrency Management Protocol

The essence of CSCWD technology is to adjust users' ideas and to execute them simultaneously without conflicts. A concurrency management protocol is used to control information that occurs concurrently in the cooperative design stage in order to guarantee an efficient access to the server and modify shared design. Borghoff and

Schlichter [7] classified it into two types: pessimistic protocol and optimistic protocol. While the pessimistic protocol consists of token-passing and locking scheme which constrains the activities of the user, the optimistic protocol focuses on high efficiency which allows multiple users to modify different nodes simultaneously. A pilgrim protocol is an example of pessimistic protocol since it uses the token passing technique using a finite state automaton model [8-10]. Garcia, *et al.* have modified the pessimistic pilgrim protocol to an optimistic pilgrim protocol using a finite state automaton approach [9]. The optimistic pilgrim protocol allows designers to modify a design model without ownership. The model is updated at the server level when a designer obtains ownership. Figure 2 shows the optimistic pilgrim finite state automaton model where pilgrim is a token in each state.

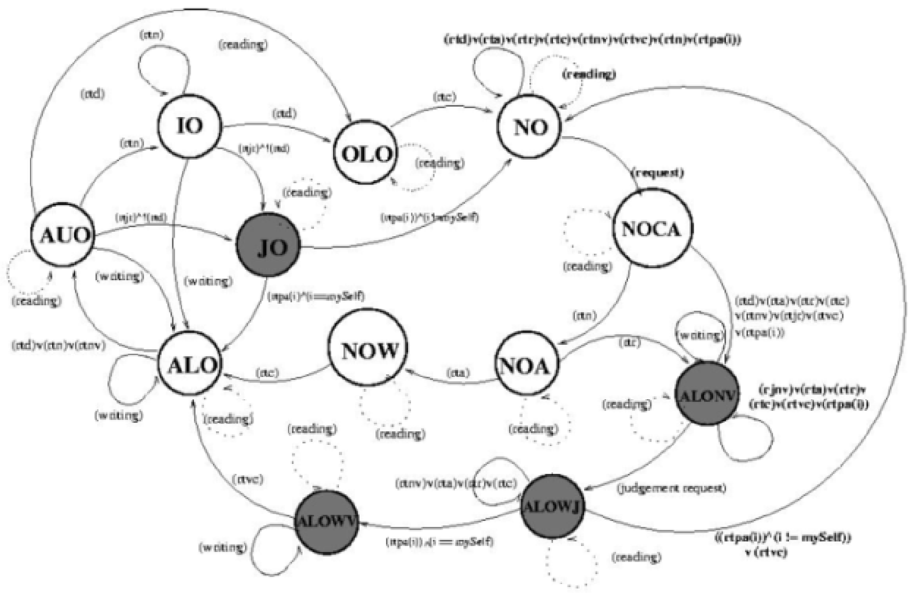


Figure 2: Garcia, *et al.*'s Optimistic *Pilgrim* Finite State Automaton [9]

While a token is just a simple variable for representing status in pessimistic pilgrim protocol, a token in optimistic pilgrim protocol has information for representing status as well as for describing ownership. The delay time for updating can be minimized using this property.

2.3 Petri Nets

A *Petri net (PN)* is one of several mathematical modeling tools. It is possible to model discrete distributed systems and verify its properties and functionalities. Petri nets can be used for many production systems as wells as CSCW systems [11]. There are different types of Petri nets: classical, timed, Stochastic and Colored Petri Nets. Classical Petri Nets is composed of places, transitions, arcs and markings. Times can be placed on transitions and places. This is classified into Timed Petri nets [12]. It is possible to represent time delays and analyze whole systems using it. Stochastic Petri nets (SPNs) [13-15] are extended timed Petri nets. It focuses on the continuous time functions and firing rules on transitions and places in order to analyze complex systems performances. Colored Petri nets (CPNs) [11, 16, 17] are a high level form of Petri nets, in which markings contain information. Information is represented as several colors. The functions are defined on transitions and conditions can be defined on arcs and each place has an associated type. An ordinary PN is defined 4-tuple (P, T, I, O) as follows,

$$\text{PN} \;\; Z = (P, T, I, O) \tag{2.1}$$

where,

P = set of places, $\{P_1, P_2, \dots, P_n\}$

T= set of transitions, $\{T_1, T_2, \dots, T_m\}$

I = set of directed arcs from P to T, $I : P \times T \rightarrow N\{0,1\}$

O = set of directed arcs from T to P, $I : P \times T \rightarrow N\{0,1\}$

Places contain any non-negative number of tokens. A transition of a Petri net may fire whenever there is a token at all its input places. After firing, it consumes these tokens, and places tokens at all its output places. Figure 3 shows a simple example of ordinary Petri net simulation.

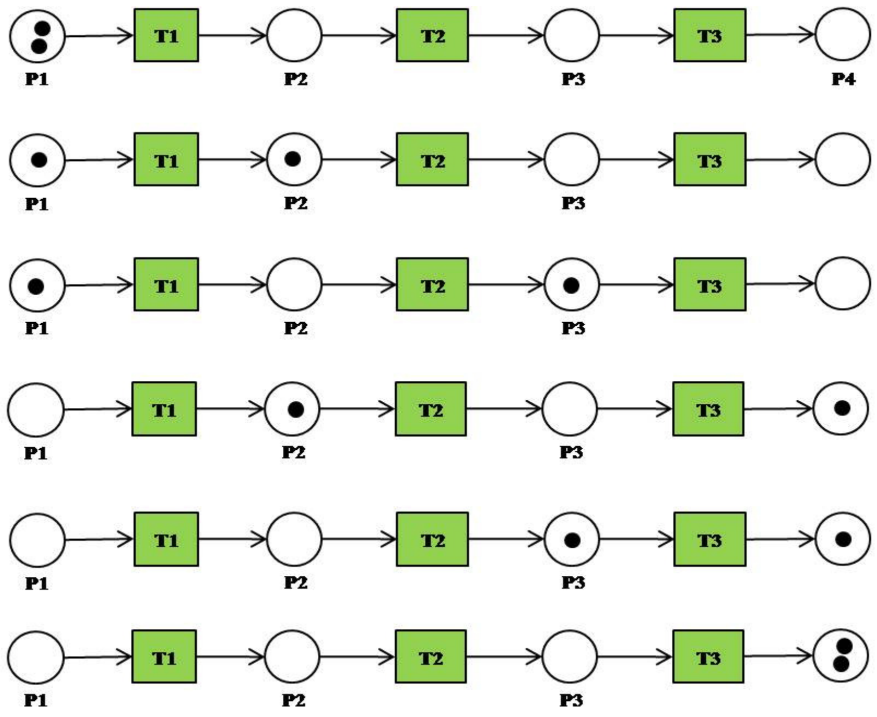


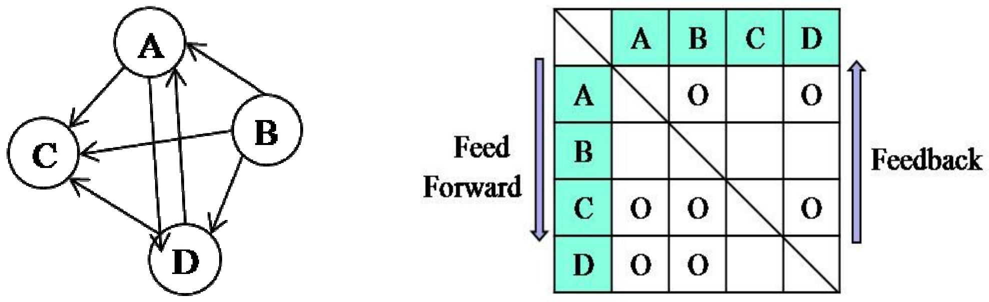
Figure 3: Ordinary Petri Net Simulation with Two Tokens

In this example, transitions are firing immediately when an input place has a token. As shown in this example, Petri nets can be used as a graphical tool. It is useful tool for describing and studying systems which are characterized as being sequential, parallel, concurrent and distributed. Including the timing and stochastic concepts into Petri net model (SPNs), complex performances and behaviors of systems can be simulated and studied. Also, CPNs provide compact models for large systems with a higher level of abstraction and an improved graphical representation capability. Generally, Petri nets are in use in a large number of areas such as communication systems, distributed algorithms, computer algorithms, flexible manufacturing systems, and many other areas. In this research, Petri Nets are used to verify concurrent management protocol and develop current protocol.

2.4 Design Structure Matrix (DSM)

DSM is a project management tool and is widely used in analyzing the planning, execution, management of complex product development projects [18]. A DSM is classified into four types: Component DSM, Team DSM, Activity DSM and Parameter-based DSM [19]. Activity DSM is used to represent the information flow between complex tasks. There are several methods to analyze DSM: partitioning, tearing and banding [18]. Activity DSM helps to manage the schedule of production efficiently since it shows the relations between time ordered activities and interdependency among the tasks. Upstream information revealed in DSM implies the iteration activities. This is useful in analyzing and tracking of the entire design process. This will be discussed in section 4.

In the graphical form of DSM representation, a node represents an activity and an edge represents the relationship between two activities. In the matrix form of DSM representation, relationships between activities are shown with a mark (O) at the corresponding location in the matrix, as shown in Figure 4 [18]. For example, if there is an edge between ith to jth node, the mark is shown at the location corresponding to column i and row j. If $i < j$, then the mark is said feed-forward; if $j < i$ then it is said feedback.

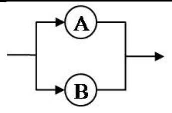
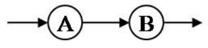
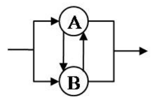
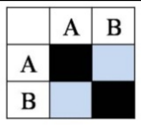
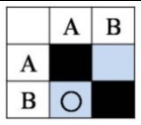
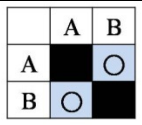


(a) Spaghetti graph (b) Base DSM

Figure 4: Spaghetti Graph of Activities and Base DSM

There are three types of activity relationships in DSM: parallel (independent), sequential (dependent), coupled (interdependent) as shown in Table 2 [18]. These configurations characterize the order of action between tasks or data exchanges. For instance, parallel indicates that there are no precedence relations to be executed prior to two activities A and B. Sequential indicates that activity A has to be preceded by activity B. Finally, coupled indicates that activity A and B influence each other. These characteristics can be used to organize several tasks systematically.

Table 2: Three Relationships in DSM

Relationships	Parallel relationship	Sequential relationship	Coupled relationship
Graph Representation			
DSM Representation			

3. COOPERATIVE MODELING ENVIRONMENT

3.1 Cooperative Modeling Architecture

In this section, the cooperative modeling architecture is described. Figure 5 shows a conceptual model of the architecture.

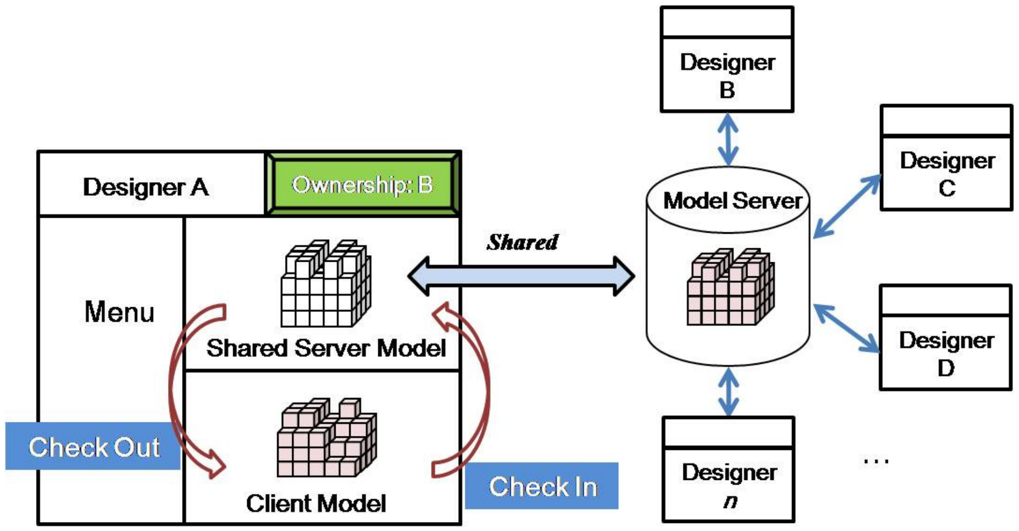


Figure 5: Cooperative Modeling Architecture

This cooperative modeling architecture consists of one model server and multiple clients. A model server has a model in the database, which is shared by each client. Each client has three types of design modules; design menu, shared server model and client model. The design menu controls activities related to the design process. It contains design libraries, parameters for feature based design and text exchanging panel. The panel of the shared server model has a copy of the model from the model server. The client model is for locally modifying each voxel model before updating the model in the model server. Even if the model is checked out by another designer, each designer can perform their updates through the client model. Once the model is checked in, the local updates are incorporated in the model at the model server. The detail mechanism is described using the revised pilgrim protocol in section 3.2.

3.2 Cooperative Modeling Protocol

3.2.1 m_token SCPNs

Before describing validation of concurrent management protocol and its development, the method to analyze this protocol is presented. As mentioned in section 2.3, Petri net is a powerful tool used in communication protocols, distributed algorithms, and many other areas. Among their application areas, communication protocols handle concurrent behaviors or problems which can occur in a cooperative work environment. Petri Net is used to model cooperative modeling protocol in order to validate behaviors. Currently optimistic pilgrim protocol is complicated to represent using simple Petri nets. Though Colored Petri net (CPN) [20] is suitable for the representation of multiple designers' behaviors, it is not enough to explain whole system and how it works. In general CPN approach, tokens have only one set of colors and transition firings are determined by the colors, whereas *pilgrim* protocol requires multiple conditions to enable transition firings. In other words, one token has to be classified into multiple sets of colors in a *pilgrim* protocol. Also transition firing time varies according to the multiple conditions of a token. To model *pilgrim* protocol based on these characteristics, a stochastic colored Petri net (SCPN) approach is required, and token have to be defined with multiple attributes. This *multi attributes stochastic colored Petri net* (m_token SCPN) is defined as follows,

$$\text{PN} \;\; Z = (\Sigma, \Lambda, P, T, C, S) \tag{3.1}$$

where,

Σ = set of color tokens with multi attributes, $\{\sigma_{1,1,1,1}, \sigma_{2,1,1,3}, \ldots\}$,

where, $\sigma_{i,j,k,l}$ = ith colored token, jth first attribute, kth second attribute, lth third attribute

Λ = set of distributions, {expo(3), 6, unif(2,3),…}

P = set of places, $\{P_1, P_2, ..., P_n\}$

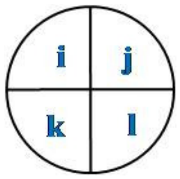
T = set of transitions, $\{T_1, T_2, ..., T_m\}$

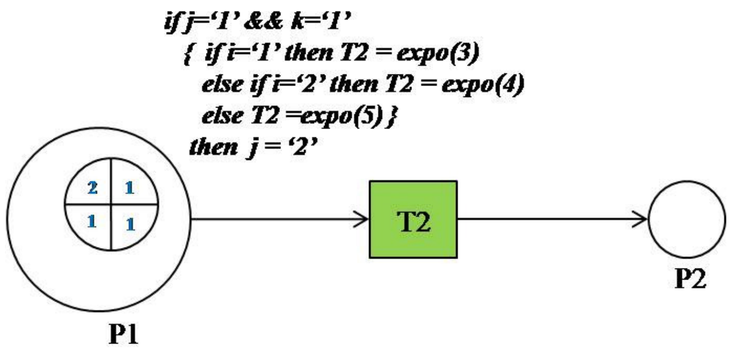
C = color function, $P \cup T \rightarrow \Sigma$

S = stochastic mapping function

$\forall t \in T : [\text{Type}(S(t)) = \Lambda \wedge \text{Type}((S(t))) \subseteq \Sigma]$

For example, if the token has four attributes: color(i) {1,2,3}, status(j) {1,2,3}, user's action (k) {1,2,3} and feasibility (l) {1,2,3}. When the token $\sigma_{2,1,1,1}$ goes to T_2, then T_2 firing time follows expo(4) and status changed to 2 (j=2) (figure 6).





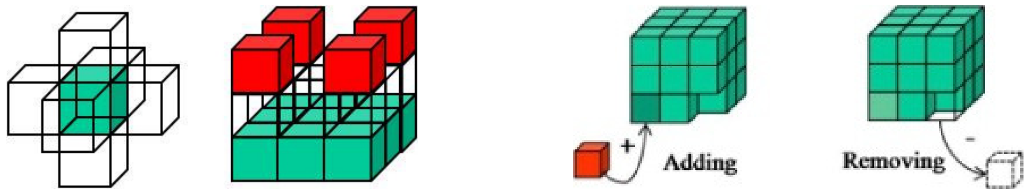
(a) m_token (b) T2 firing condition and its time

Figure 6: m_token SCPN Example

As shown in the above example, a transition can fire whenever all its attributes satisfy the required conditions. Some of the attributes may change after the transition firing. m_token SCPN methodology is useful for modeling designer's type, behaviors and their changing status in the system. Also, it is useful for performance analysis and evaluation of a system based on variety of transition firing times.

3.2.2 Revised Optimistic pilgrim protocol

The optimistic pilgrim protocol is used as a basic concurrency design protocol. As discussed in section 2.2, under the optimistic pilgrim protocol, a user obtains ownership of a model using a token prior to performing any updates on a model residing in the server. While the owner edits the model, other users can implement their changes to a client model and the data is stored locally using a token. After the owner returns the ownership, another user can gain ownership which allows the update of the server model with any existing client model updates. However this protocol has some limitations. First, the optimistic pilgrim protocol focuses only on transmitting the data updates to the server without determining any inconsistency problems that might arise due to design constraints and infeasibilities. Disjointed or degenerated voxel are two cases of infeasible design that can occur. Disjointed voxel is shown in Figure 7(a) where red colored voxels are separate from the main body which is shown using green colored voxels. Degeneracy can be defined as adding a voxel at a location already containing a voxel or removing a voxel from a position that does not contain a voxel. This is illustrated in Figure 7(b).



(a) Disjointed voxel

(b) Degeneracy of voxel

Figure 7: Infeasible Voxel Design

Second, this protocol does not consider designers or client's type (e.g. dominant designer, reviewer) which leads to it not providing sufficient opportunity to dominant designer. If the ownership is given to a dominant designer prior to a reviewer, the design process can be performed more effectively. To overcome these two limitations, a revised optimistic pilgrim protocol is proposed with m_token SCPN methodology. In this protocol, a token has four attributes and those are shown in table 3.

Table 3: Multi Attributes of Token Used in Revised Optimistic Pilgrim Protocol

Attributes	Declarations
s (shape)	type SHAPE = {circle, plus, box} var s = SHAPE
a (action)	type ACTION = {'request' , 'check_out' , 'writing' , 'correction'} var a = ACTION

f (feasibility)	type FEASIBILITY = {'disjoint' , 'degeneracy' , 'feasible'}
	var f : FEASIBILITY;

t_s
(token status)
	type TOKEN STATUS = {'no_com', 'accepted' , 'refused', 'changed'}
	var t_s : TOKEN STATUS;

Figure 8 shows the procedure of infeasibility control in the system. A designer has ownership which corresponds to Active Owner (AO) place. If a disjointed voxel is generated, designer A transitions from AO place to *Active Owner with Infeasible Design* (AO-IFD) place, and the designer has to check his/her action: infeasible or feasible. If the designer judges that it is a feasible action, then designer transitions from AO-IFD to AO place. Or if the designer judges that it is not feasible action and modifies it correctly, then the designer transitions from AO-IFD place back to AO place. If degeneracy is detected, designer state transitions from AO to *Active Owner with Degeneracy* (AO-D) place. This results in a warning message to the designer. If the designer wants to correct it then designer state transitions from AO-D to AO place.

Figure 9 shows how this protocol manages multiple designers given the priority rule. In this example, there are one dominant designer (s='circle') and two reviewer (s='plus' or 'box') participating in design process. All the tokens are gathered in P9 and sorted by its SHAPE type. If token shape is circle, it transitions to Dominant Non-owner which wins the ownership (D_NOW) place. If the token is plus or box, they transition to Reviewer Non-owner which wins the ownership (R_NOW) place respectively. There are two management rule: (i) dominant has priority over reviewer, and (ii) First Comes First Serve (FCFS). If circle and plus tokens are present in P10 and P11 simultaneously, T19 can fire ahead of T20 when token status (t_s) is equal to 'changed'. The Inhibitor arc between P10 and T17/T18 enables this priority procedure. If P9 has a plus or a box, any token who arrives to P11 or P12 first, has a priority since the resource to firing T17 or T18 is limited to one (P14). Using this protocol, the model server controls multiple clients, prevents design conflict, and minimizes waiting time for access ownership. The whole structure of original/revised pilgrim protocol PNs and the comparison result of two protocol's process time are shown in figures 18-22 which are included in Appendix A.

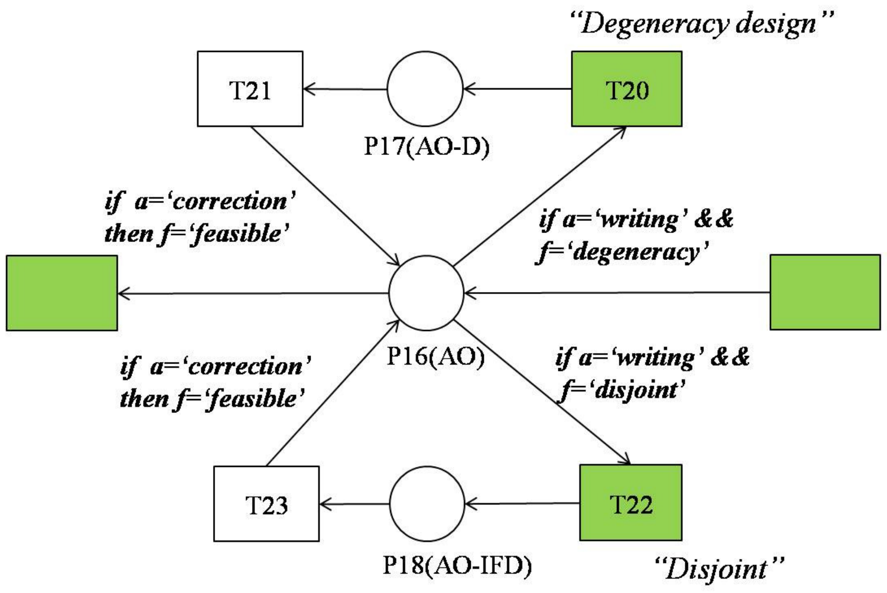

"Degeneracy design"
T21
T20
P17(AO-D)
if a='correction' then f='feasible'
if a='writing' && f='degeneracy'
P16(AO)
if a='correction' then f='feasible'
if a='writing' && f='disjoint'
T23
T22
P18(AO-IFD)
"Disjoint"

Figure 8: Feasibility Checking Function

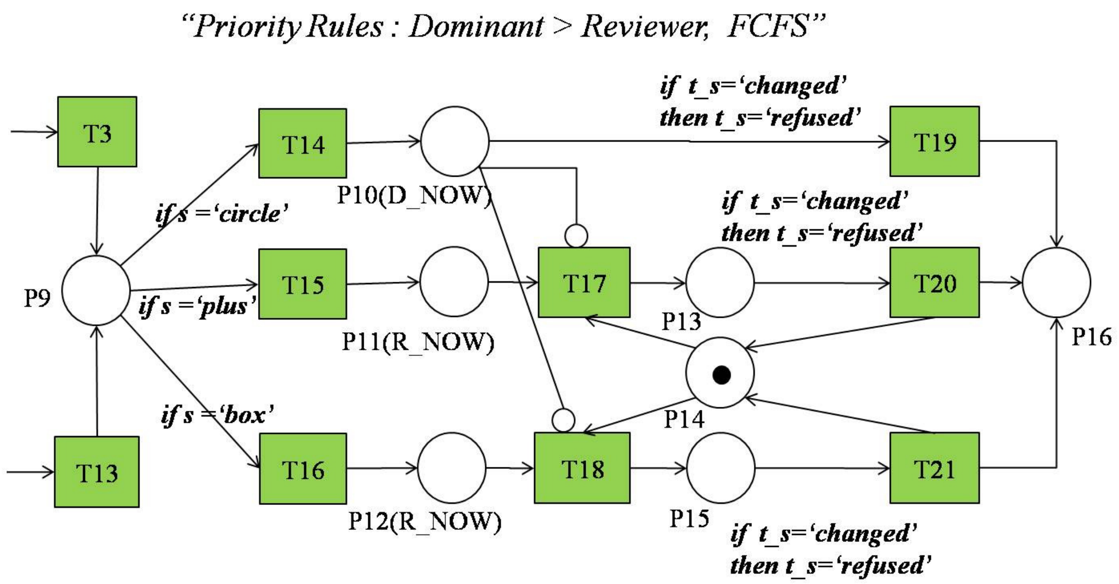

"Priority Rules : Dominant > Reviewer, FCFS"
T3
T14
P10(D_NOW)
if t_s='changed' then t_s='refused'
T19
if s ='circle'
P9
T15
if s ='plus'
P11(R_NOW)
T17
P13
if t_s='changed' then t_s='refused'
T20
P16
P14
if s ='box'
T13
T16
P12(R_NOW)
T18
P15
T21
if t_s='changed' then t_s='refused'

Figure 9: Management of Multiple Designers by Priority Rule

4. ALGORITHM FOR DESIGN HISTORY TRACKING

In many existing CAD software such as *CATIA*, *I-DEAS*, and *AutoCAD*, one of the major limitations is the limited amount of design history tracking and its applications. Even though there have been efforts devoted to storage and retrieval of design information, these are mainly focused on archiving product data and knowledge management. In this research, the goal is to track design data at every step and use it to obtain an optimal design path and identify/remove redundancy using a Design Tracking Matrix (DTM).

4.1 Voxel-Based Modeling Procedure

The voxel-based modeling procedure and its simple algorithm are first described. Building or modification of a shape is performed through the addition or removal of voxels. As shown in the Design Node steps in Figure 10, voxel data is being added ("+" mark) or removed ("-" mark) to obtain the final shape (7th node). Additions occur in chronological order of node number while removals occur in reverse order in the graph. Addition represents feed-forward while removal represents feedback edge between two nodes (i and j) and they are defined as the transition function (δ) of feed-forward and feedback as follows,

$$FW_{i,j} = \delta(i, j \mid j \neq \phi) \quad \forall i < j \tag{4.1}$$

$$FB_{j,i} = \delta(j, i \mid i \cap j \neq \phi) \quad \forall j < i \tag{4.2}$$

Two conditions must be satisfied for these transition functions:

- Feed-forward function: destination node is not empty
- Feedback function: there is an intersection between ith and jth nodes

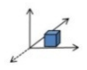
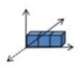
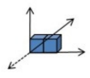
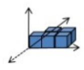
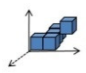
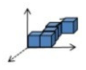
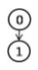
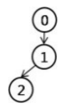
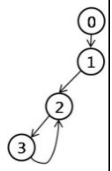
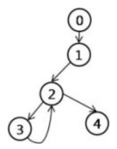
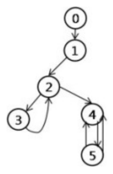
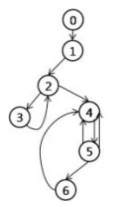
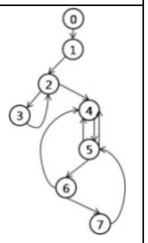
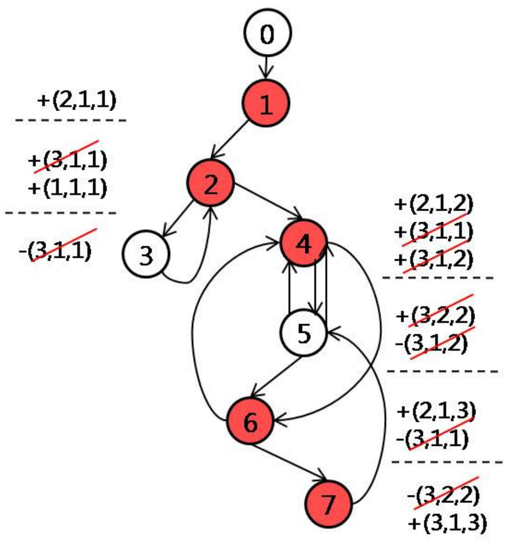
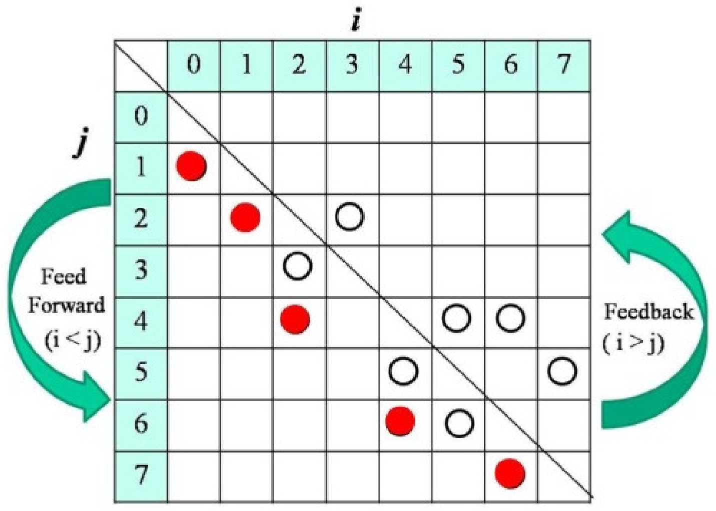
Design Node	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Shape								
Design Tracking Graph								
Voxel Data	·	+ (2,1,1)	+ (3,1,1) + (1,1,1)	- (3,1,1)	+ (2,1,2) + (3,1,1) + (3,1,2)	+ (3,2,2) - (3,1,2)	+ (2,1,3) - (3,1,1)	- (3,2,2) + (3,1,3)

Figure 10: Voxel-Based Design Process and Its Graph

Figure 11(a) describes the design graph in more detail. Each node contains voxel data which is created by designers and the edge shows the addition or removal of voxel data between nodes.



(a) Design tracking graph



(b) Design Tracking Matrix (DTM)

Figure 11: Design Tracking Graph and Its DTM

4.2 Design Tracking Matrix (DTM)

Design Tracking Matrix (DTM) is modified from Design Structure Matrix (DSM). DTM focuses on representing data flows between design steps. The main difference between the two methods is that partitioning the DSM rows and columns is unnecessary in DTM. Partitioning is the process of manipulating the matrix in order to eliminate or reduce the feedback mark in DSM. The partitioning is not needed in DTM, since the feedback mark represents an important design process as it is. When a design step is represented in chronological order (ex.1→2→3) the feedback mark shows the reverse order process such as 3→2 or 3→1. To track the optimal design path, these processes must be followed through unchanged and in the same time order. A designer can determine redundant steps and trace optimal design path using the data exchange captured in DTM.

Based on this design tracking graph, a DTM can be obtained as shown in Figure 11(b). Marks (O) indicate an edge's start/end nodes and the margin of matrix shows the design nodes. The feedback marks appear in the upper diagonal matrix, while feed-forward marks appear in the lower diagonal. The design data exchange can be captured using these sets of two marks. The DTM expressions can be represented as follows:

$$D = [d_{ij}]_{m \times n} \tag{4.3}$$

$$ud_{ij} = \{d_{ij} \mid \forall i, j; i > j\} = \{0,1\} \tag{4.4}$$

$$ld_{ij} = \{d_{ij} \mid \forall i, j; j > i\} = \{0,1\} \tag{4.5}$$

where

m, n = dimensions (number of graph nodes)

i, j = ith and jth design node

ud_{ij} = Upper Diagonal mark (element)

ld_{ij} = Lower Diagonal mark (element)

4.3 Optimal Design Path and Final Shape

4.3.1 Optimal Design Path

Given a DTM, the optimal design path can be obtained. The optimal design path is the collection of nodes in columns which contain at least one voxel data. It is the same as the set of final nodes in every column located in the lower diagonal mark. The lower diagonal represents the addition of voxels and final nodes contain voxel data in every column. From equation (4.1), if there are no voxels in following nodes, feed-forward function cannot be activated. Equation (4.6) describes the representation of optimal design path (OP).

$$OP = \{ j \mid d_{\min(i),j}, d_{i,\max(j)}, ld_{ij} = \{1\} \} \qquad \forall i, j \;\; j > i \tag{4.6}$$

where

$d_{\min(i),j}$ = the mark located at the min position for all jth design nodes

$d_{i,\max(j)}$ = the mark located at the max position for all ith design nodes

In Figure 11, the colored nodes indicate the optimal design path. To prove equation (4.6), we assume that 5th node is an element of the optimal design path. According to the OP definition, the 5th node has to contain at least one voxel data. In that case, the transition between 4th node to 6th node cannot occur, since the 5th node is not empty. This contradicts the transition condition made in the feed-forward transition function. According to the condition of feedback function there must be an intersection between following and preceding node to execute it. To satisfy this condition under the current assumption, one of voxels cannot belong to their nodes : +(3,1,2) at 4th node or -(3,2,2) at 7th node. This is the second inconsistency. Therefore, optimal design path formulation is verified.

4.3.2 Final Shape

By choosing all the elements in the optimal design path, the final shape of a model can be constructed using the voxel value at every location. It can be stated as

$$S(i,j,k) = \{(i,j,k)_p \mid \forall i \in I, \forall j \in J, \forall k \in K, p = 1\} \tag{4.7}$$

where

$$(I,J,K) = \left\lceil \frac{\text{Total length of design model in X(, Y, Z) axis}}{\text{unit length of a voxel}} \right\rceil$$

$$(i,j,k)_p = \text{particular voxel}$$

$$\text{p} = \begin{cases} 1 & \text{if voxel exist in final shape} \\ 0 & \text{otherwise} \end{cases}$$

5. ALGORITHM FOR DESIGN ANALYSIS

5.1 Numerical Design Tracking Matrix (NDTM)

In the DTM, the mark (O) provides a single binary attribute which signifies the existence or absence of relationship between two design nodes. A Numerical DTM (NDTM) uses a number instead of a mark to represent multiple attributes, such as redundancy and degree of usage. Redundancy can be characterized as the removed voxel data which is not contained in the final shape but is present in the shared server model during the design process. Degree of usage (DU) is the ratio of the existent voxel data to all of the generated voxel data related to the particular design node. Redundancy is represented in the upper diagonal matrix while degree of usage is shown in the lower diagonal matrix. They can be obtained by

$$U_k = \frac{n_k - r_k}{n_k} \tag{5.1}$$

$$r_k = \sum_{i=k+1}^{m} ud_{i,k} \tag{5.2}$$

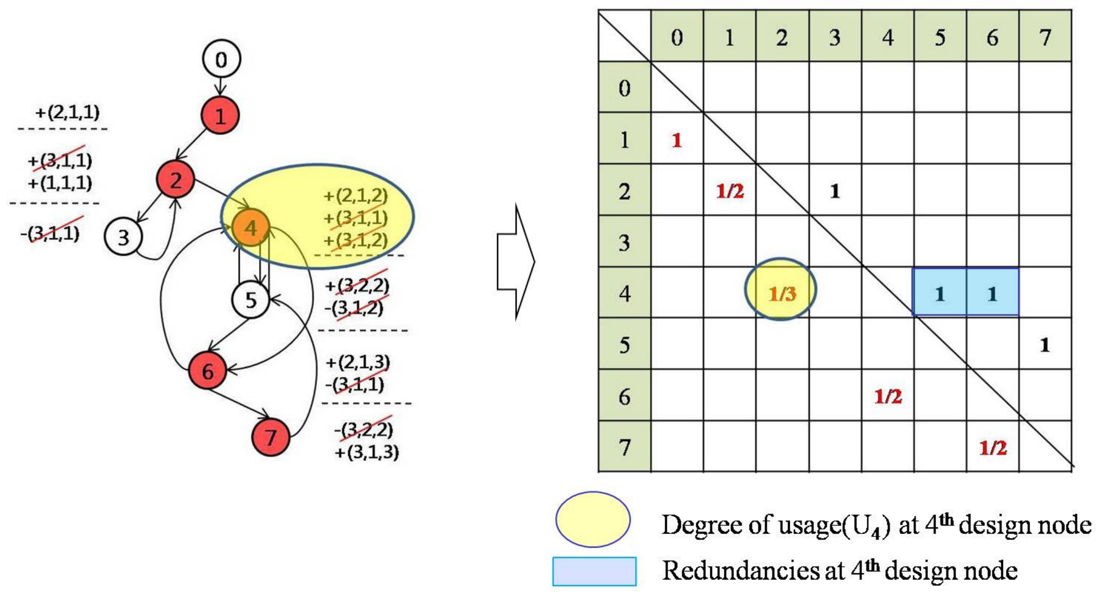
where

U_k = degree of usage at k^{th} node

n_k = number of whole voxel data which are generated at k^{th} node

r_k = number of redundancies at k^{th} node

Figure 12(b) shows the NDTM with degree of usage and redundancy which is derived from the design tracking graph (Figure 12(a)). Degree of usage at 4th design node (U_4) is equal to 1/3 since three voxels are generated and two of them are removed later. In this example, number of redundancies at 4th design node is equal to 2.



(a) Design tracking graph (b) NDTM with degree of usage and redundancy

Figure 12: NDTM with Its Attributes

5.2 Similarity and Dissimilarity Measure

In the past, many cooperative activity models have focused on the efficiency and concurrency of the work and provide an environment to organize multidisciplinary user's performance well [1-3]. They neglect to study the characteristics of participants or their design pattern. An algorithm is presented to identify similarity or dissimilarity among designers' modeling patterns. To identify similarity or dissimilarity among designers' modeling patterns, it is necessary to understand the condition of comparison and property of degree of usage.

- Condition: Compare design differences that occur sequentially
- Property of DUs: Preceding NDTM has higher value of DUs than following NDTMs'

First, the comparison is limited to contiguous two design cases since the proposed design method is processed in a sequential manner: Owner designer updates his/her design to model server, followed by the modifications to the design at the model server by the next designer. This implies that the following NDTM is generated from the previous NDTM. So these two NDTMs will be compared. Second, it is possible that the NDTM being used by the second designer contains modifications to eliminate redundancies from the previous design modifications. In such a case, the modified NDTM will have a smaller DU as compared to the previous NDTM.

Given a NDTM, the difference between design A and B (D_{AB}) is formulated as,

$$D_{AB} = \frac{\sum_{k=l}^{m}(|U_{Ak} - U_{Bk}| \cdot w_k)}{w_k} \qquad 0 \le D_{AB} \le 1 \qquad (5.3)$$

where

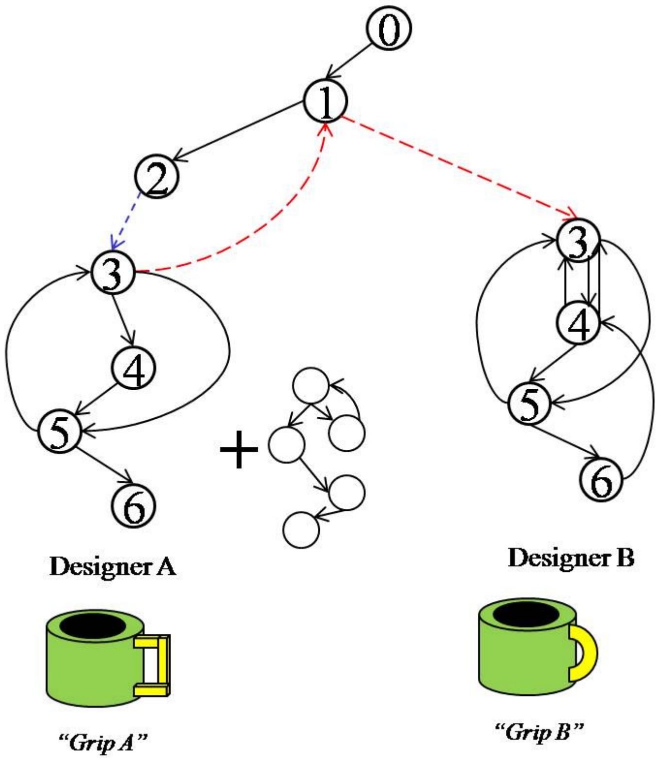
$l =$ first node to compare

$m =$ final node to compare

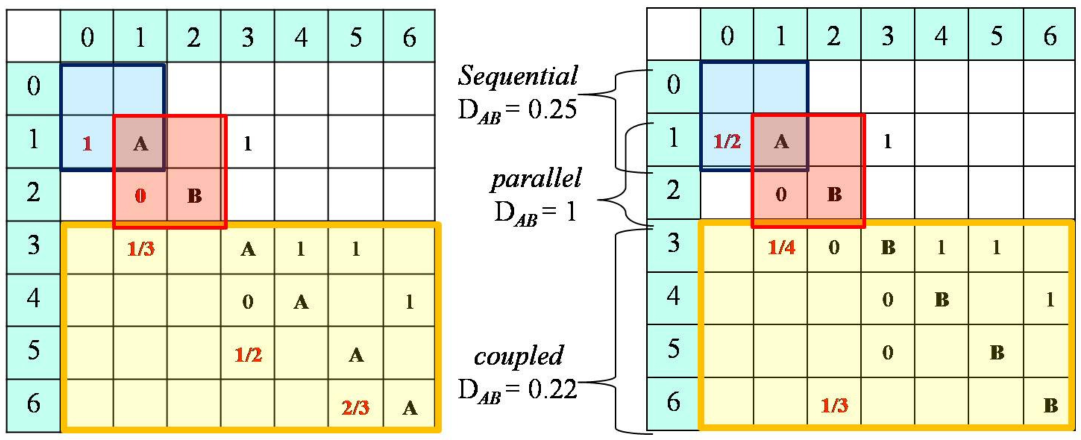
$k = k^{\text{th}}$ design node

$w_k =$ number of generated voxel at k$^{\text{th}}$ design node

When D_{AB} value is close to zero, there is little difference between the two designs. $D_{AB} = 1$ indicates a totally different design. As a result of comparison between design A and B, three main relationships (similar to the parallel, sequential and coupled relationships mentioned in section 2.4) can be developed. In a NDTM, relationships are classified as parallel, sequential and coupled similar to the independent, dependent and interdependent relationships in DSM. Figure 13(b) shows an example of difference between two designs. Parallel and sequential relationships are shown in 2nd and 3rd design node while a coupled relationship is shown in multiple node comparison (4th to 6th). Table 4 shows the coupled case comparison between two designs. The compared part (cup's grip) is designed by designer A, and modified by designer B later as shown in Figure 13(a).



(a) Design change from designer A to Designer B


Sequential
D_{AB} = 0.25
parallel
D_{AB} = 1
coupled
D_{AB} = 0.22
Designer A's NDTM
Designer B's NDTM

	0	1	2	3	4	5	6
0							
1	1	A		1			
2		0	B				
3		1/3		A	1	1	
4				0	A		1
5				1/2		A	
6						2/3	A

	0	1	2	3	4	5	6
0							
1	1/2	A		1			
2		0	B				
3		1/4	0	B	1	1	
4				0	B		1
5				0		B	
6			1/3				B

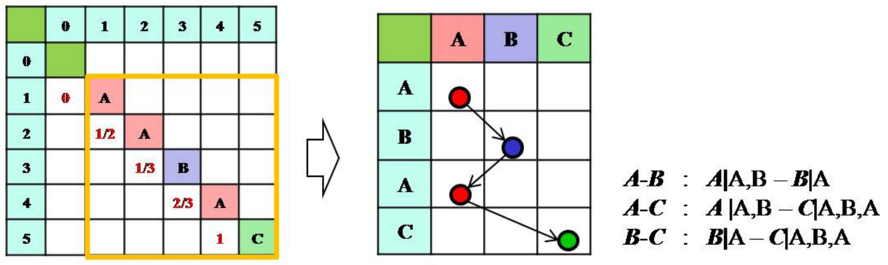
(b) Three similarity relationships between two designers

Figure 13: Design Comparison between Designer A and B

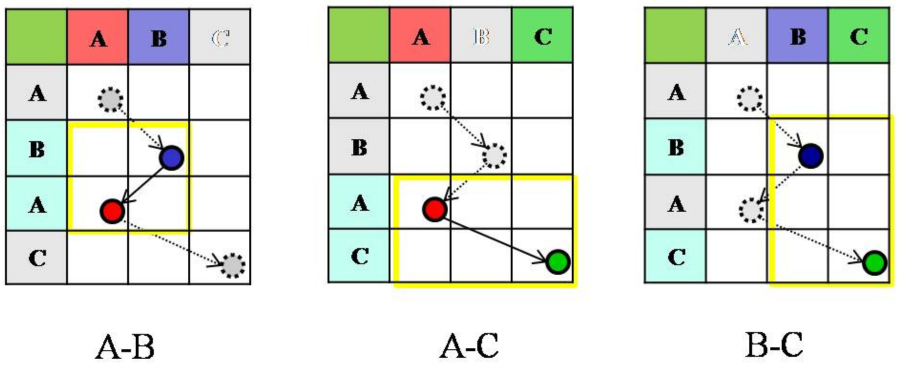
Table 4: Coupled Case Comparison

k	U_{Ak}	U_{Bk}	$\lvert U_{Ak}-U_{Bk} \rvert$	W_k	$\lvert U_{Ak}-U_{Bk}\rvert \cdot W_k$
4	1/3	1/3	0	3	0
5	*0*	0	0	1	0
6	1/2	0	1/2	2	1
7	2/3	1/3	1/3	3	1
				D_{AB}	0.222

Also, this similarity/dissimilarity comparison can be applied to three more designers. Based on the DTM, comparison matrix can be obtained like as figure 14(a). It is not necessary to track all the design iterations to compare the three designers since design tracking matrix contains whole design changes from start to the point of comparison time. As shown in figure 14(b), 1st design iteration which is designed by A is not necessary to compare designer A and B (D_{AB}).



(a) Comparison matrix obtained from 5 iteration NDTM



(b) Comparison set

Figure 14: Multiple Comparison Matrix and Comparison Set

5.3 Participating Portion with Octree Approach

An octree subdivision method is required to represent the voxel-based design model in detail. Octree subdivision is the process of dividing an initial voxel into octant and subdividing it into sub-octants [21]. In our model architecture, each designer picks up the necessary voxel and subdivides it repeatedly until the desired model is obtained. Under this design process, each designer contributes to the final shape of the design model partially. However, it is difficult to estimate the number of voxels or designs generated by each designer accurately, since the design process becomes complicated through the subdivision method. In order to obtain each designer's contribution to the whole design, it is necessary to distinguish the steps of each design and participants.

At the first octree design level (level 0), one NDTM is generated for all the designers' work. After that, they choose one voxel and start the iterative process of breaking it down for the detail representation. In this design step, multiple NDTMs are generated and the octree design level is one. Every subdivided voxel has its own NDTM in every octree level until the final shape of design model is acquired. Figure 15 shows this octree subdivision design structure with NDTMs.

Before figuring out each designer's contributions towards the whole design, in a NDTM, the designers' participating portions (e.g. if there are n designers – $P_1, P_2, \ldots, P_n$) are expressed as

$$P_i = \frac{(\sum_{k=0}^{j} n_k - r_k)_i}{\sum_{k=0}^{j} n_k - r_k} \quad , i = 1,\ldots,n\text{-}1 \tag{5.4}$$

$$P_n = 1 - \sum_{i=1}^{n-1} P_i \tag{5.5}$$

where

i = ith designer

j = final design node

$\sum_{k=0}^{j} n_k - r_k$ = number of the existent voxels in a task

$(\sum_{k=0}^{j} n_k - r_k)_i$ = number of the existent voxels which are generated by designer i

In Figure 15, there are five voxels generated by designer A, B and C respectively at the first octree design level (level 0). In this example, the participating portions are $P_A = \frac{3}{5}$, $\mathrm{P_B} = \frac{1}{5}$ and $\mathrm{P_C} = \frac{1}{5}$.

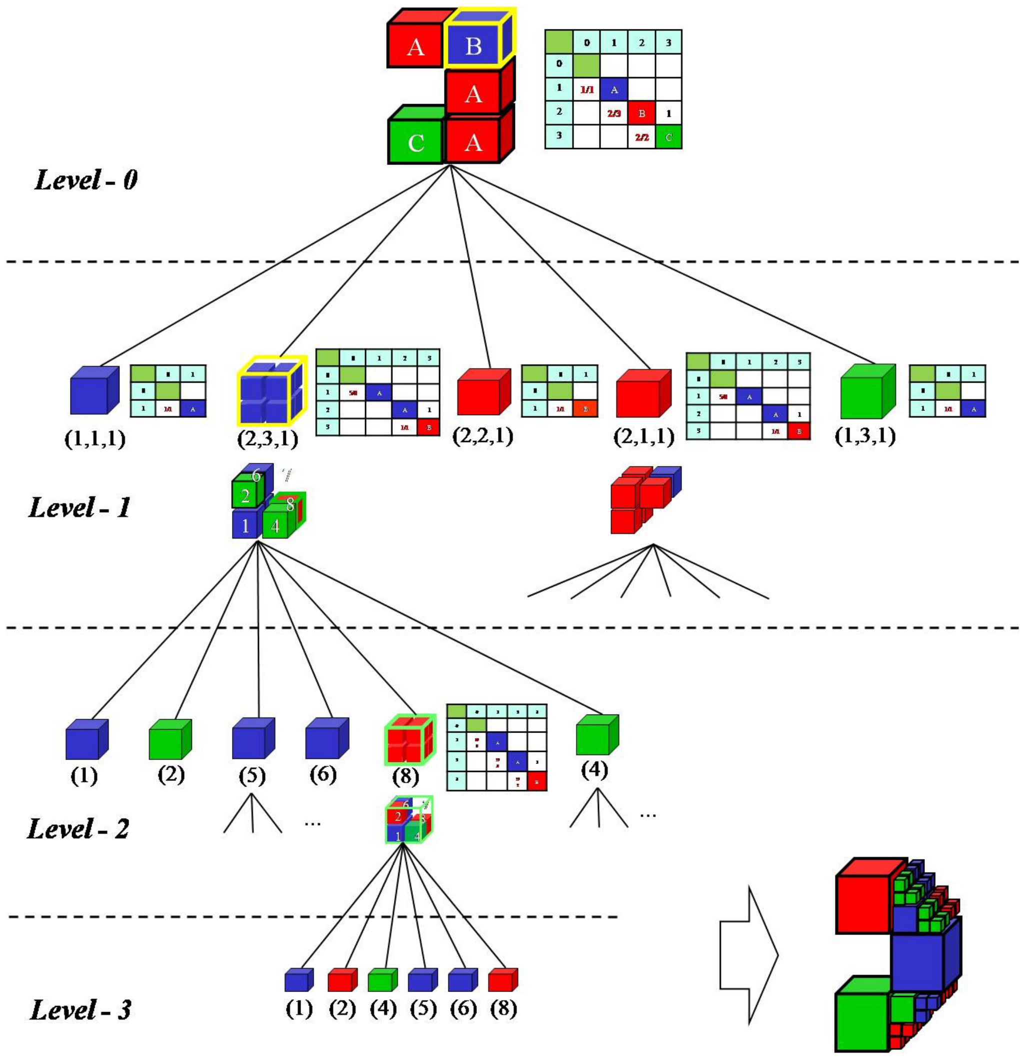


Figure 15: Octree Subdivision Design Structure with NDTMs

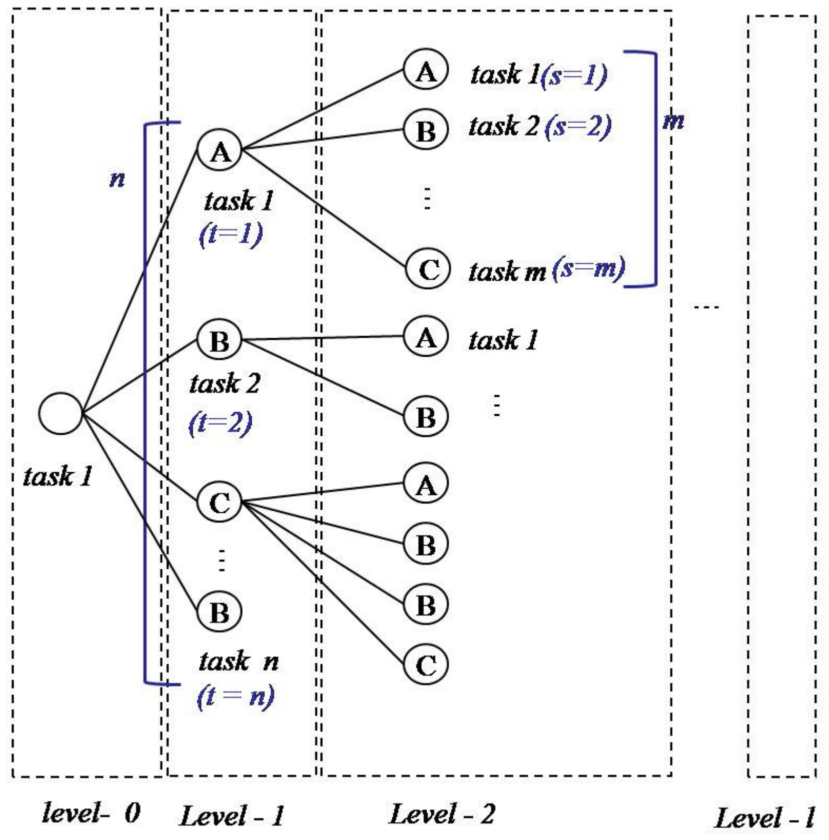


Figure 16: Voxel-Based Octree Subdivision Design Structure

Voxel-based octree subdivision design structure can be represented as shown in figure 16. In this thesis, each NDTM is defined as a task at each octree level. During the octree subdivision design process, if a particular participating portion in *t*th task at the *l*th level is changed, it can be obtained by,

$$P_{i,(l,t)} = \frac{\sum_{s=1}^{m} P_{i,(l+1,s)}}{(\sum_{k=0}^{j} n_k - r_k)_{(l-1)}} \qquad s = 1,2,\ldots,m \quad \text{and} \quad l = 1,2,... \tag{5.6}$$

$$P_{i,(0,1)} = \sum_{t=1}^{n} P_{i,(1,t)} \qquad t = 1,2,\ldots,n \tag{5.7}$$

where

$i = i^{th}$ designer

l = octree design level

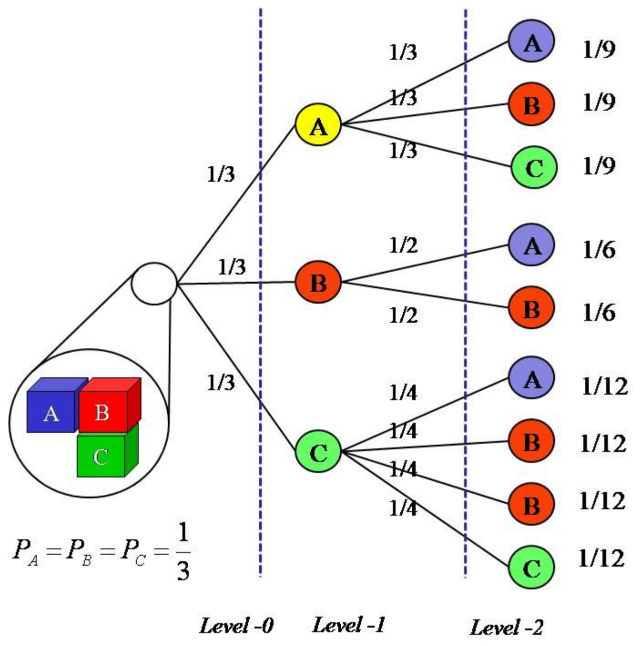
t = task number which assigned to octree design level l

s = task number in octree design level $i+1$ at task t

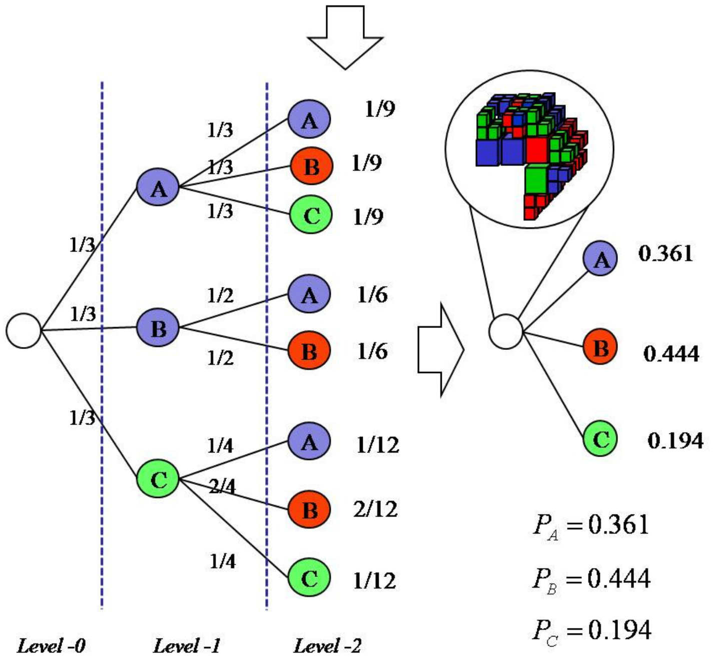
$P_{i,(l,t)}$ = particular portion in t^{th} task at l^{th} level for designer i

Figure 17 shows the participating portion changes with octree subdivision graph. In this graph, the nodes represent the tasks at each octree design level and the edge indicates the subdivision of particular voxel. The edge number represents the portions about how many sub-voxels are assigned to the following octree design level. The right end value of this graph shows the participating portions.

While equation (5.7) is the formula to obtain the designer's contributions toward the whole design, equation (5.6) gives the value for the representation of merging same nodes. For example, yellowed node A (figure 17(a)) is the first task at the octree design level-1 and its $P_{A,(1,1)}$ is equal to 2/9. This is the same as the right end value where two nodes and edges are merged into one as shown in Figure 17(b). Through the merging of same nodes and edges, the graph is simplified and finally P_A and P_B are computed.



(a)



(b)

Figure 17: Participating Portion Changes with Octree Subdivision Graph

6. RESEARCH SUMMARY AND CONCLUSION

6.1 Research Contribution and Conclusion

The proposed voxel-based cooperative design architecture supports effective cooperative design functions. First of all, it provides an effective cooperative design environment using revised optimistic pilgrim protocol. This protocol enables minimizing delay time for updating design works, preventing possible design inconsistency, and managing multidisciplinary designers effectively. Second, it provides a method to obtain the optimal design path using DTM. By the process of eliminating unnecessary design steps, a complicated design process can be simplified and standardized design path can be obtained for a product design. Standardized design paths can be stored in a design library and utilized in similar product design scenarios in the future. Third, it helps capture modeling pattern such as adding preference or deleting preference, whole to small part or small to whole part. Modeling pattern represents each designer's specialized approach associated with product design and it can be used for analyzing designer's performance and utilized in future modeling work. Similarity/dissimilarity measure enables finding the differences between each designer's modeling patterns. Finally, the overall redundancies obtained from NDTM can be used to evaluate designer's design efficiency. Those functions are not provided in existing cooperative CAD systems. As opposed to existing CAD systems which focus on simple history changes, the proposed algorithm provides an efficient method to track, access and manage history.

6.2 Future Scope of Study

In this research, only voxel based representation is considered. However many CAD software use B-rep representations or their own native representation methods. These representations can also be captured by design tracking graph and DTM. In this case, the content of design tracking graph and DTM will be determined by the operators and parameters used in the CAD system. Ultimately, it will be possible to develop a system which will predict and provide feasible design alternatives based on the tracked database. This system will allow easy and fast product design.

REFERENCES

[1] M. L. Maher and J. H. Rutherford, "A Model for Synchronous Collaborative Design Using CAD and Database Management," *Research in Engineering Design,* vol. 9, pp. 95-88, 1997.

[2] R. Bidarra, E. V. D. Berg, and W. F. Bronsvoort, "A collaborative feature modeling system," *Journal of Computing and Information Science in Engineering,* vol. 2, pp. 192-198, 2002.

[3] W. D. Li, J. Y. H. Fuh, and Y. S. Wong, "An Internet-enabled integrated system for co-design and concurrent engineering," *Computers in Industry,* vol. 55, pp. 87-103, 2004.

[4] J. J. Shah and M. Mantyla, *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications*. New York: Wiley, 1995.

[5] V. Chandru, M. Manivannan, and S. Manohar, "Minkowski operator and feature of voxel models," in *ASME Design Engineering Technical Conferences*, Las Vegas, 1999, pp. 1-11.

[6] V. Chandru and S. Manohar, "Volume modeling for emerging manufacturing technologies," *Sadhana,* vol. 22, pp. 199-216, 1997.

[7] U. M. Borghoff and J. H. Schlichter, *Computer-Supported Cooperative Work Introduction to Distributed Applications*. New York: Springer, 2000.

[8] H. Guyennet, J.-C. Lapayre, and M. Trehel, "The pilgrim: A new consistency protocol for distributed shared memory," in *Proceedings of the ICA3PP'97 International Conference*, Melbourne, Australia, 1997, pp. 253-264.

[9] E. Garcia, J.-C. Lapayre, and G. David, "Pilgrim Performance over a New CAlif Communication Layer," in *Proceedings of 7th Parallel and Distributed Systems International Conference (ICPADS'00)*, Iwate, Japan, 2000, pp. 203-210.

[10] E. Garcia, H. Guyennet, J. Henriet, and J.-C. Lapayre, "Towards an Optimistic Management of Concurrency: A Probabilistic Study of the Pilgrim Protocol," in *Computer Supported Cooperative Work in Design II: 9th International Conference*, Coventry, UK, 2005, pp. 51-60.

[11] L. M. Kristensen and L. Petrucci, "An Approach to Distributed State Space Exploration for Coloured Petri Nets," in *25th Int. Conf. Application and Theory of Petri Nets (ICATPN'2004)*, Bologna, Italy, 2004, pp. 474-483.

[12] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ: Prentice Hall PTR, 1981.

[13] S. U. Guan and S. S. Lim, "Modeling Adapatable Multimedia and Self-Modifying Protocol Execution," *Future Generation Computer Systems,* vol. 720(1), pp. 123-143, 2004.

[14] S. M. Koriem, T. E. Dabbous, and W. S. El-Kilani, "A new Petri Net Modeling Technique for the performance Analysis of Discrete Event Dynamic Systems," *Journal of Systems and Software,* vol. 72, pp. 335-348, 2004.

[15] G. L. Park, "Performance Evaluation of a List Scheduling Algorithm in Distributed Memory Multiprocessor Systems," *Future Generation Computer Systems,* vol. 20(2), pp. 249-256, 2004.

[16] K. Jensen, "Colored Petri Nets and the Invariant Method," *Theorical Computer Science,* vol. 14, pp. 317-336, 1981.

[17] K. Jensen, "An Introduction to the Practical Use of Coloured Petri Nets," *Lectures on Petri Nets II: Applications LNCS,* vol. 1492, pp. 237-292, 1998.

[18] A. Yassine, "An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method," in *Quaderni di Management (Italian Management Review)*, 2004.

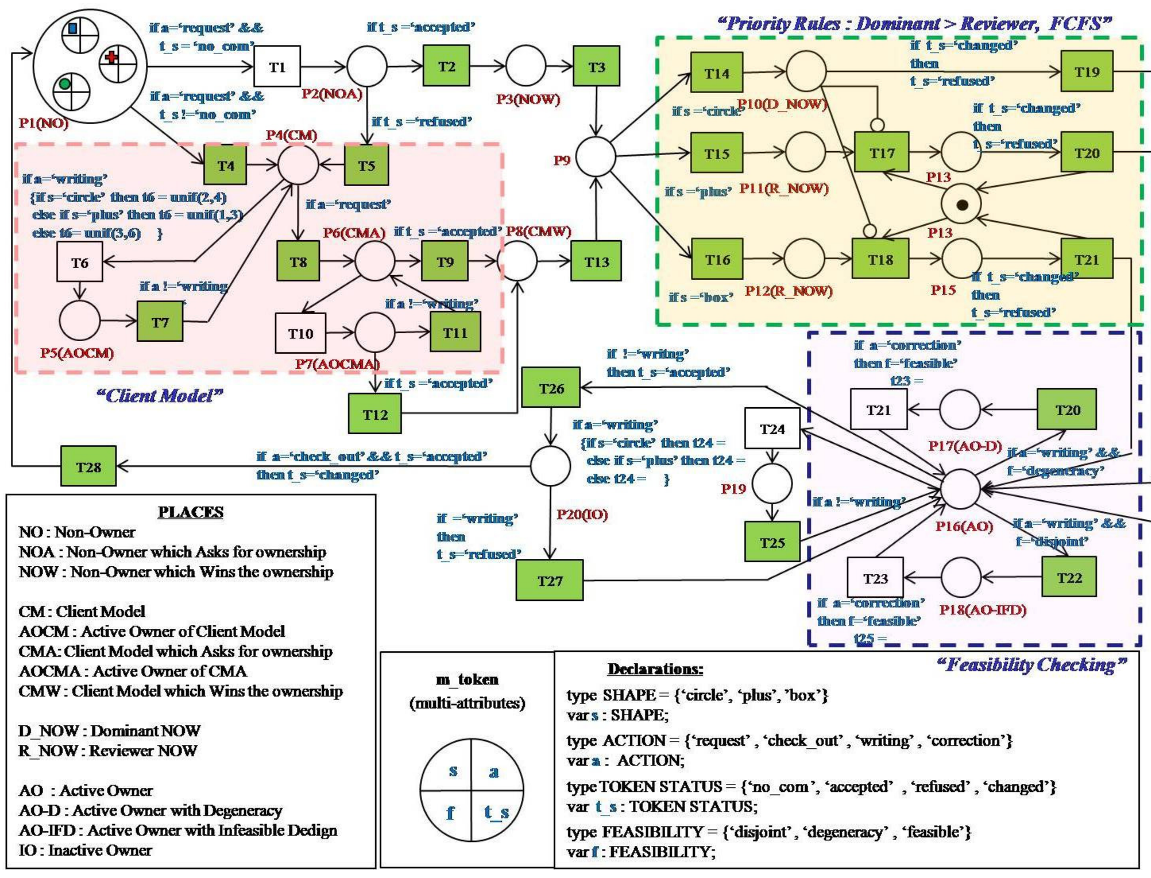
[19] T. R. Browning, "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions on Engineering Management,* vol. 48, pp. 292-306, 2001.

[20] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for Modeling and Validation of Concurrent System," *Int J Software Technology Transfer,* pp. 213-254, 2007.

[21] Y.-J. Tseng and Y.-R. Sue, "Machining of Free-Form Solids Using an Octree Volume Decomposition Approach," *International Journal of Production Research,* vol. 37, pp. 49-72, 1999.

APPENDIX A

REVISED OPTIMISTIC PILGRIM PROTOCOL



"Client Model"
"Priority Rules : Dominant > Reviewer, FCFS"
"Feasibility Checking"
P1(NO)
P2(NOA)
P3(NOW)
P4(CM)
P5(AOCM)
P6(CMA)
P7(AOCMA)
P8(CMW)
P9
P10(D_NOW)
P11(R_NOW)
P12(R_NOW)
P13
P15
P16(AO)
P17(AO-D)
P18(AO-IFD)
P19
P20(IO)
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
T16
T17
T18
T19
T20
T21
T22
T23
T24
T25
T26
T27
T28
if a='request' && t_s = 'no_com'
if a='request' && t_s != 'no_com'
if t_s ='accepted'
if t_s ='refused'
if a='writing' {if s='circle' then t6 = unif(2,4) else if s='plus' then t6 = unif(1,3) else t6= unif(3,6) }
if a='request'
if a !='writing'
if t_s ='accepted'
if s ='circle'
if s ='plus'
if s ='box'
if t_s='changed' then t_s='refused'
if a='correction' then f='feasible' t23 =
if a='correction' then f='feasible' t25 =
if a='writing' && f='degeneracy'
if a='writing' && f='disjoint'
if a !='writing'
if !='writng' then t_s='accepted'
if a='writing' {if s='circle' then t24 = else if s='plus' then t24 = else t24 = }
if a='check_out' && t_s='accepted' then t_s='changed'
if ='writing' then t_s='refused'

PLACES

NO : Non-Owner
NOA : Non-Owner which Asks for ownership
NOW : Non-Owner which Wins the ownership

CM : Client Model
AOCM : Active Owner of Client Model
CMA: Client Model which Asks for ownership
AOCMA : Active Owner of CMA
CMW : Client Model which Wins the ownership

D_NOW : Dominant NOW
R_NOW : Reviewer NOW

AO : Active Owner
AO-D : Active Owner with Degeneracy
AO-IFD : Active Owner with Infeasible Dedign
IO : Inactive Owner

m_token
(multi-attributes)

s	a
f	t_s

Declarations:

type SHAPE = {'circle', 'plus', 'box'}
var s : SHAPE;
type ACTION = {'request' , 'check_out' , 'writing' , 'correction'}
var a : ACTION;
type TOKEN STATUS = {'no_com', 'accepted' , 'refused' , 'changed'}
var t_s : TOKEN STATUS;
type FEASIBILITY = {'disjoint' , 'degeneracy' , 'feasible'}
var f : FEASIBILITY;

Figure 18: Revised *Pilgrim* Protocol SCPNs with m_token

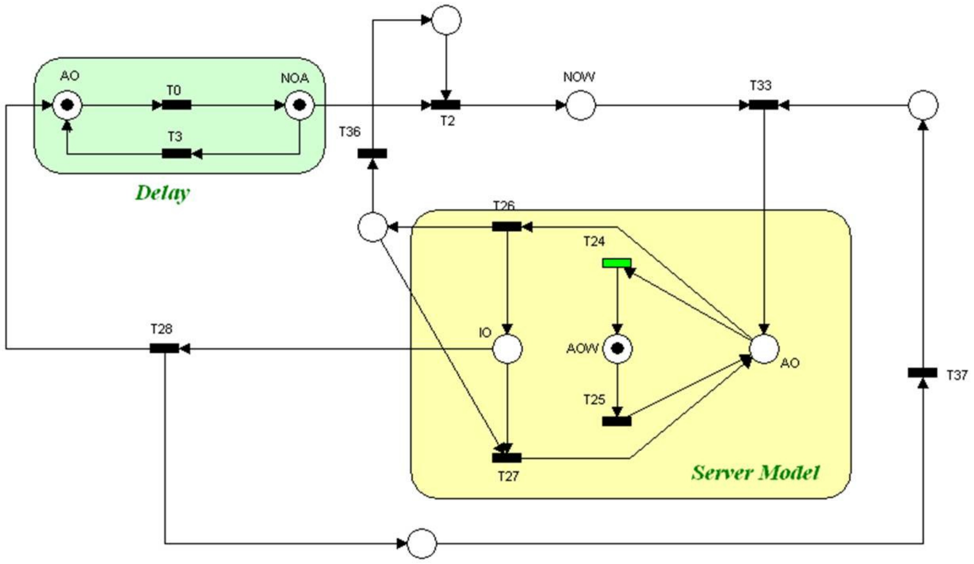


Figure 19: Original Pessimistic *Pilgrim* Protocol with Hpsim Simulation

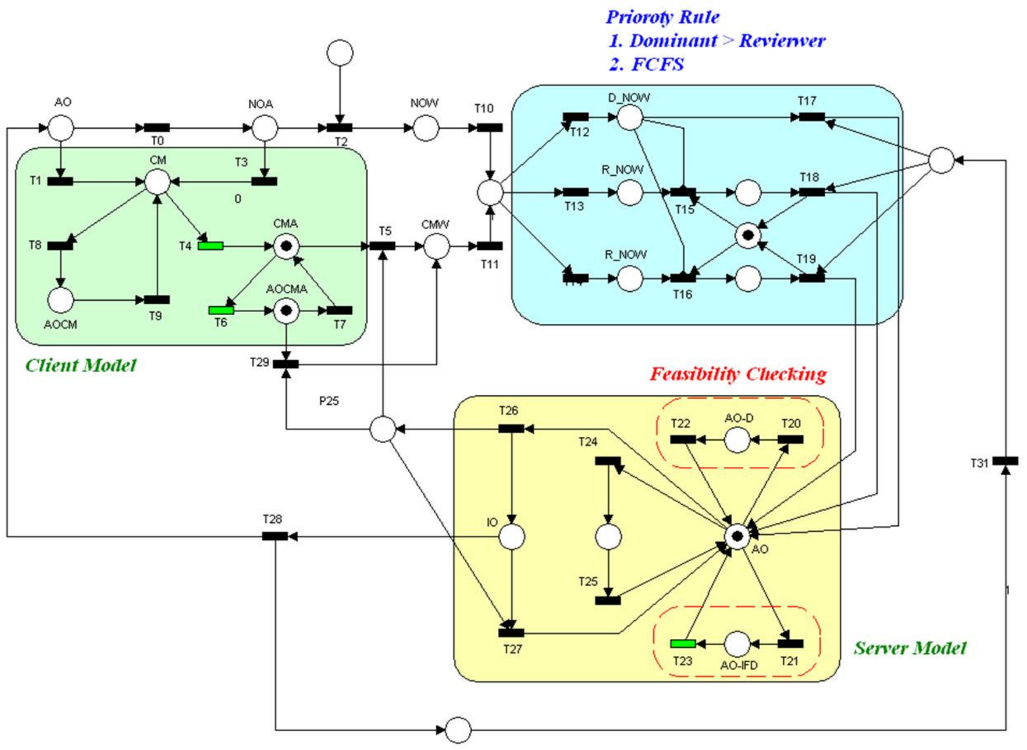
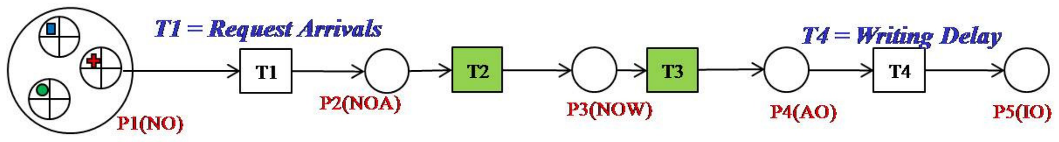
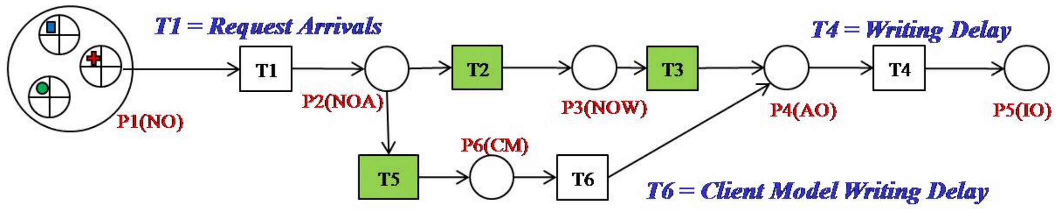


Figure 20: Revised Optimistic *Pilgrim* Protocol with HPsim Simulation



(a) Origin *pilgrim* protocol



(b) Revised *pilgrim* protocol

Figure 21: Simplified Model of Two Protocols

Simulation assumptions to compare the process times of two protocols are as follows,

(i) All transitions fire immediately except T1, T4 and T6

(ii) All updating times are zero in server model

(iii) All correction times for infeasible designs are zero

(iv) All time delays use the data in table 5

(v) Any token which consumes time delay in T6 has different delay time in T4 (T4' = T6 –T4)

(vi) Total process time is sum of whole token's delay time until final token arrives in P5

(vii) The time unit is minutes

Table 5: Stochastic Time Delays in *Pilgrim* Protocol

		Original '*pilgrim*'			Revised '*pilgrim*'			
Type		'circle'	'plus'	'box'	'circle'		'plus'	'box'
Request Arrivals	T1	exp(5)	exp(3)	exp(4)	exp(5)		exp(3)	exp(4)
Client Model Writing	T6	.	.	.	U(2,4)		U(5,7)	U(4,6)
Server Model Writing	T4	N(10,4)	N(12,3)	N(8,2)	T4	N(10,4)	N(12,3)	N(8,2)
					T4'	N(10,4) - U(2,4)	N(12,3) - U(5,7)	N(8,2) - U(4,6)

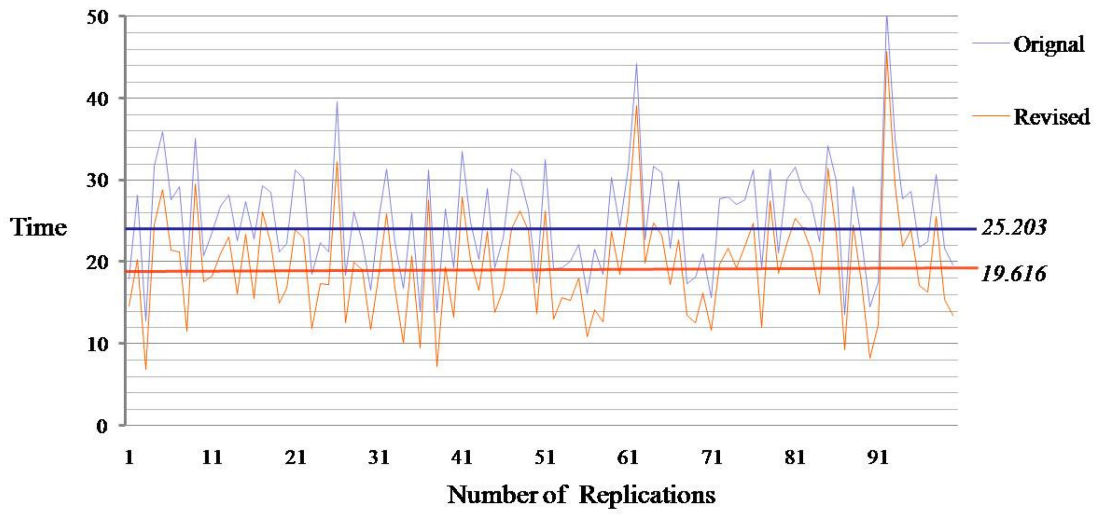


Figure 22: Total Process Times of Two Protocols

Table 6: Simulation Results with 100 Replications

Protocol	Average time	Min Time	Max time
Original ‘*pilgrim*’	25.203	12.787	50.610
Revised ‘*pilgrim*’	19.616	6.8305	45.728

VITA

Jonghyun Kim was born in Taegu, South Korea. He received his Bachelor's degree in architecture engineering in 2004 from Korea Military Academy in Seoul. He was commissioned as a second lieutenant in 2004 and serves in the Army as a military officer up to now. He joined the industrial engineering graduate program at Texas A&M University in September 2007 and graduated with his M.S in August 2009. He is married to Mijae Kim and has a son, Gio Kim. His permanent address is: 2014-212, Daemyeoung-dong, Namgu, Taegu, South Korea. His email is: lop3538@gmail.com.