Coordination of Distributed New Product Development Processes for Multiple Design Chains

Shyh-ming Lin1, Ching-hung Tseng2

Department of Industrial Management, National Pingtung University of Science and Technology, 1, Shuefa Road, Neipu, Pingtung 912, TAIWAN R.O.C.

1smlin@mail.npust.edu.tw; 2smooth870318@gmail.com

Abstract- This study addresses the conflict problem of DNPD (Distributed New Product Development) schedules that arise among multiple design chains. A DNPD process is composed of certain parallel and sequential activities, fulfilled by a core company and some satellite companies. Companies involved in DNPD processes form a loosely coupled multiple design chains network. The design chain companies, in dynamically changing business environment, must frequently adjust their ongoing DNPD activities to meet market requirements. Any adjustment made in the multiple design chains network is likely to bring the interdependent companies into conflict. With no coordination mechanism, such kind of conflict will delay the product launch time, lead to considerable profit loss, and make the core companies lose their competitive positions. This study tries to solve the conflict problem by first adopting the congestion games model to formulate the design chain dynamic conflict problem, proposing an agent based problem solving technique to coordinate the conflict and then applying physical agents to implement a distributed coordination mechanism for the multiple design chains. To evaluate the performance of the mechanism, this study takes the simulated multiple design chains as the cases to carry out some experiments on it. The experimental results demonstrate the validity of our mechanism.

Keywords- Design Chain; Distributed New Product Development; Process Schedule Conflict Coordination; Congestion Games; Software Agents

I. INTRODUCTION

NPD (New Product Development) is the most important stage in product life cycle that NPD determines 80% of product cost [1]. GGI (Goldense Group, Inc.) in 2002 in their survey found that 42% of companies give priority to new products over existing products when allocating resources [2].

The NPD process is formed by four major phases: ideas generation and conceptual design, definition and specification, prototype and development, and commercialization [3]. In those phases, several activities are to be executed concurrently or sequentially to launch a new product. NPD activities are always distributed over multiple organizations [4]. A survey in 2006 reported that 29% of U.S. manufacturing companies had outsourced part of their new product development process and 41% had considered outsourcing [5]. Another survey in 2008 said that 65% of U.S. based electronics manufacturers had outsourced their new product development activities [6]. The drivers of outsourcing are cost saving, revenue increasing, skilled and experienced people available, quality improving, and time to market shortening [5-9].

The outsourcing trend leads companies to form design chains in their DNPD (Distributed NPD) processes and to embrace open innovation [7, 8, 10]. The companies encompass cross-functional interdependent organizations, such as product designers, marketers, parts suppliers, manufacturers, customers. A design chain requires the exchange and coordination of different skills and knowledge [11-12]. The management of a design chain is to direct all the organizations, including some satellite company and a core company, to contribute their knowledge and experience to DNPD [11].

The DNPD is defined as the separation and optimization of activities performed during a single product development process across multiple geographic locations [13]. In a DNPD process, a core company acts as the leading role and some satellite companies serve as engineering service suppliers to support the core company. A core company and its satellite companies are loosely-coupled, i.e., they have no direct parent-child relations and their relationships are temporary and weakly interdependent, beginning from the time an NPD process is formed and ending when the process is fulfilled. The loosely-coupled relation enables the satellite companies to join in more than one design chain and facilitates the multiple design chains to operate simultaneously in a distributed design chains network.

In the increasingly unpredictable and dynamic business environment, outsourcing NPD work subjects companies to significant uncertainty [14]. The uncertainty is likely to bring the dynamic conflicts to the distributed design chains network if limited resources are requested or expected to be reallocated. The conflicts among design chain activities will delay the time-to-market or result in the project failures [15]. A Mckinsey study indicated that when a company ships products 6 month late, the loss will be 33% of its after-tax profit [16]. If a new product cannot be launched either ahead of time or on time, it follows that the core company will not be able to gain first mover advantage and market share, nor will it be able to make huge profits.

The reasons why uncertainty in design chains network will bring dynamic conflicts are as follows. (1) The organizations own only finite service resources and cannot unlimitedly satisfy all design chain requests; for example, the manpower of internal or external organizations is restricted, and so are the capacities of test or simulation facilities. (2) No organization is willing to share with others
its private information (for example, the status of capacity loading or which customer at what time places an order), and no one can obtain the information about all the design chains. (3) The organizations are self-interested and bounded rational [17]. Every organization pursues its personal optimization instead of global optimization subject to available information, and, (5) centralized coordination mechanism is remotely possible for the context of inter-organizational coordination is needed [8]. The design chain coordination in recent years has been generating great research interest [8, 18-20]. However, the conflict coordination across multiple distributed design chains network has never been studied.

To boost the DNPD performance, infrastructure for inter-organizational coordination is needed [8]. The design chain coordination in recent years has been generating great research interest [8, 18-20]. However, the conflict coordination across multiple distributed design chains network has never been studied.

This study focuses on solving the dynamic conflict problem in a decentralized design chains network by proposing a distributed coordination mechanism. To build the mechanism, the congestion games model is adopted to formulate the design chain conflict problem, an agent based coordination method is proposed and then physical agents are applied to implement the coordination mechanism.

This paper is organized as follows. Section 2 reviews the related works. Section 3 introduces an abstract example to demonstrate how the dynamic conflicts arise and how to coordinate the conflicts among multiple design chains. Section 4 develops the coordination method and mechanism for design chains network. Section 5 designs and carries out experiments to evaluate the mechanism performance and discusses the experimental results. Finally in Section 5, this study is concluded and some future research directions are proposed.

II. RELATED WORKS

The related works, including the congestion games model, multi-objective optimization problem and Pareto improvement, and multi agent technology, are reviewed as follows.

A. Congestion Games

Congestion games were first introduced by Rosenthal in 1973 [21]. Rosenthal modeled the scenario of the congestion games as that n selfish players plan their min-cost path independently to travel from certain origins to certain destinations in a capacity-limited road network. Vöcking and Aachen thought that congestion games lie in the intersection between optimization and game theory [22] because each player in the game seeks for his/her optimal solution, i.e., minimum length, among a set of feasible paths that link specified sources and destinations. Besides, in the congestion game, the players simultaneously seek at maximizing their individual payoffs and the payoff depends on the choices of all players. The characteristics of the congestion games imply that it can be used to formulate a game problem in which multiple rational players compete for limited resources. The congestion game has been extended to many applications [23-27].

A congestion game (CG) can be expressed by a 4-tuple equation CG=<N, R, (Σi)i∈N, (di)i∈N> [23]: N the set of game players, R the set of limited resources, Σi ⊆ 2R the strategy space of player i, and di the cost function corresponding to the resource r. S = (S1,.....Sn) is a state of the game in which player i acts selfishly and aims at minimizing his/her cost while adopting strategy Si, subject to limited resources and other players’ strategies S−i. The aforementioned characteristics show that the congestion game has potential to model the design chains conflict problem.

Vöcking and Aachen further defined the improvement of a congestion game solution as follows [22]. Under any state S = (S1,.....Sn), a player i decreases his/her cost by changing his/her strategy from S to S′.

Solving a congestion game is hard [28]. Heuristic algorithms are now the major problem solving techniques to congestion games [29].

B. Multi-Objective Optimization Problem and Pareto Improvement

Multi objective optimization problem (MOOP) is an optimization problem with multiple objective functions which aims at simultaneously optimizing more than one objective. To solve a multi-objective optimization problem is to find a number of Pareto -optimal solutions, rather than find a single, globally optimal solution. Pareto-optimality, named after Vilfredo Pareto, is the outcome of a game under the condition that there is no other outcome that makes every player at least as well off and at least one player strictly better off. The Pareto optimality E and weak Pareto optimality WE are defined as follows [30]:

Pareto optimality E: Consider K conflicting objectives Zi to be minimized. S is the set of solutions and Z its image in the objectives space. x ∈ S is a Pareto optimum (or an efficient solution) iff ¬∃y ∈ S, x ≠ y, such that Z(y) ≤ Z(x), ∀i, i=1,.......,K, with at least one strict inequality. E is the set of all Pareto optima and ZE its image in the objectives space.

Weak Pareto optimality WE: Consider K conflicting objectives Zi to be minimized. S is the set of solutions and Z its image in the objectives space. x ∈ S is a weak Pareto optimum (or a weak efficient solution) iff ¬∃y ∈ S, x ≠ y, such that Z(y) < Z(x), ∀i, i=1,.......,K. WE is the set of all Pareto optima and ZWE its image in the objectives space. One has E ⊆ WE.

Pareto improvement is an act to reallocate resources and the reallocation makes at least one player better off without making anyone else worse off. An allocation of resources is Pareto efficient or Pareto optimal when no further Pareto improvements can be made.

Pareto improvement in this study is used to establish a decision rule built into the physical agents for core companies to autonomously resolve the design chain
dynamic conflicts and then improve their DNPD schedules.

C. Multi Agents Technologies

Agents are intelligent software programs that serve as personal assistants performing certain duties. Agents are autonomous, proactive, and adaptive software that can perform complicated tasks autonomously to solve a growing number of complex problems [31-35]. Besides, agents contain some level of intelligence, from fixed rules to learning engines that allow them to adapt to changes in the environment [36]. Russell and Norvig [37] completely characterized the agent as follows: an agent operates under autonomous control, perceives their environment, persists over a prolonged time period, adapts to change and is capable of taking on another’s goals.

A multi-agent system is a combinatorial software system that is made up of more than one agent. In a multi-agent environment, agents can either cooperate with other agents to carry out a common duty or compete with other agents to pursue their own interests. Durfee and Lesser [38] defined the multi-agent system as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver. Jennings et al. [39] identified the characteristics of multi-agent systems as: (1) each agent has incomplete capabilities to solve a problem; (2) system control is distributed; (3) data are decentralized; (4) computation is asynchronous.

Multi agents, which can interact with each other in physical environment without boundary, are called physical agents. Physical agents, on behalf of the individuals, can communicate, coordinate, and collaborate with corresponding physical agents representing other individuals.

The characteristics of physical agent technologies described here present their abilities to support multiple design chains coordination in distributed environment. The multi-agent systems can help core companies to plan their individual DNPD schedules, reserve required services by coordinating with satellite companies, improve their DNPD schedules by coordinating with other companies to resolve the possible occurring conflicts and adjust themselves to reduce the impact made by market changes or environmental uncertainties.

In this study, JADE serves as the agent platform for coordination mechanism. JADE (Java Agent DEvelopment Framework, http://sharon.cselt.it/projects/jade/) is a physical multi-agent development framework which complies with FIPA specifications and aims at simplifying the implementation of multi-agent systems. It includes two main products: a FIPA-compliant agent platform and a package to develop Java agents. The advantages of utilizing JADE as a physical agent platform include its capabilities to distribute agents across different operating platforms and to make these agents easily communicate with each other through the Internet.

The efforts of developing physical agents as software systems are firstly summarized as distributed artificial intelligence in Bond and Gasser’s book [40], and the interoperability of multi-agent system across the Internet had been initiated by the Foundation for Intelligent Physical Agents (FIPA, http://www.fipa.org) in 1996. FIPA was formed to produce software standards for heterogeneous agent-based systems. Fig. 1 illustrates the building layers of physical agents complying with FIPA standards (http://www.fipa.org). In Fig. 1, the FIPA abstract architecture specifications describe the abstract entities that are required to build agent services and environments. FIPA agent communication specifications define Agent Communication Language (ACL) messages, message exchange interaction protocols, speech act theory-based communicative acts and content language representations. FIPA agent management specifications define the control and management of agents within and across agent platforms. FIPA agent message transport specifications define the transport and representation of messages across different network transport protocols, including wire and wireless environments.

Based on the FIPA specifications shown in Fig. 1, Mitkas shared his experiences that developing the JADE application requires the following four duties: (1) creating agent ontologies, (2) creating behavior types for agents, (3) creating agent types, and (4) deploying the multi agent system [41].

III. DEMONSTRATION OF AN ABSTRACT DESIGN CHAIN CONFLICT PROBLEM

This section introduces an abstract example to clearly demonstrate how the conflicts occur in a distributed design chain network and how the conflict can be resolved. Based on such idea, in the next section, a coordination method is developed and a coordination mechanism is implemented.

Fig. 2 shows the involved organizations in a design chain network scenario in which P1, P2 and P3 are the core companies and F1, F2 and F3 serve as the satellite companies to help P1, P2 and P3 fulfill their DNPD projects. P1, partnering F1, F2 and F3, forms one design chain; P2,
partnering F_2 and F_3, forms another design chain and P_3, partnering F_1, F_2, forms the other design chain.

Core companies

P_1

P_2

P_3

F_1

F_2

F_3

Satellite companies

Fig. 2 Design Chain Network Scenario

The processes of P_1, P_2 and P_3’s DNPD projects are shown in Fig. 3. In Fig. 3, p_11, f_1, f_2, f_3 and p_12 are the activities in P_1’s process; p_11 and p_12 are the internal activities of P_1, and f_1, f_2 and f_3 the activities outsourced to F_1, F_2 and F_3. P_2 and P_3 are similar to P_1. In Fig. 3, the tables, below the process diagram of P_1, P_2 and P_3, are the estimated time required to complete every activity.

In Fig. 3, assumptions are made as follows:

• Core companies P_1, P_2, P_3 and satellite companies F_1, F_2, F_3 are rational, in that their objectives are to maximize individual interests. A core company’s objective is to schedule its DNPD process and to fulfill the schedule as early as possible whereas a satellite’s is to raise its capacity utilization as high as possible, namely, to shorten as much idle time as possible.

• P_1, P_2, P_3 and F_1, F_2, F_3 can only engage in one activity at one time. That is, their resources are limited and renewable (the resource availability is specified for each period).

• F_1, F_2, F_3’s resources are irreplaceable.

• Every activity in a process must be carried out and fulfilled in one whole time period to make the activity result complete. For example, activity f_2 can be carried out continuously during 4 to 5, but not in separate time 4 and 7.

• Organizations in the network maintain private information and can access limited information by querying the available service time of satellite companies. Apart from this, any organization in the environment is not able to obtain the detailed information of others. Under such circumstances, conflicts are likely to occur when core companies need satellite companies’ services to achieve individual objectives.

• In the real world, core companies simultaneously schedule their DNPD processes and compete for the services of satellite companies. However, in this example to clearly describe how core companies’ schedules are optimized, P_1, P_2 and P_3 are supposed to sequentially enter the design chain network and orderly request for F_1, F_2 and F_3’s services. Namely, P_1 is the first entrant who has the first priority to reserve satellite companies’ service time. P_2 the second and P_3 the last.

P_1

P_2

P_3

F_1

F_2

F_3

Activity | Time required
---|---
F_1 | 3
f_1 | 6
f_2 | 2
f_3 | 4
P_1 | 3
Activity | Time required
---|---
P_2 | 2
f_1 | 3
f_2 | 2
P_2 | 3
Activity | Time required
---|---
P_3 | 3
f_1 | 4
f_2 | 2
f_3 | 4
P_3 | 4

Fig. 3 DNPD Projects in Design Chain Network

The following four subsections explain how the conflicts among design chains will occur and how the conflicts are resolved and the joint optimization reached.

A. Scheduling for DNPD Processes Individually

This subsection presents the course of how core companies schedule their DNPD processes, query and reserve the available service time of satellite companies, and how core companies communicate with satellite companies.

Based on the assumptions made in Fig. 2 and 3, P_1 arranges its schedule as shown in Fig. 4. The schedule in Fig. 4 is optimal for P_1. After arranging the schedule, P_1 sends feasible time query messages to F_1, F_2, and F_3 respectively to query about the earliest beginning time of f_1, f_2 and f_3. F_1, F_2, and F_3, after receiving the queries, check their capacity loading. Since the queried time periods are idle, F_1, F_2, and F_3 then respectively reply with feasible time response messages to P_1. Fig. 4 shows that P_1 will spend 16 time units to fulfill its DNPD project.
Fig. 4 P1’s Optimal Schedule

Fig. 5 shows P1’s self arranged optimal schedule without considering the resource limitations of satellite companies. Based on the schedule in Fig. 5, P2 sends feasible time query message to F2 and F3 to query about their available time of f2 and f3. F2 and F3, after receiving the queries, check their capacity loading respectively. For F2, since the expected beginning time is occupied, a conflict will occur. F2, after knowing the conflict, tries to find the earliest available time period for P2 and replies with a feasible time response to P2. Receiving the response, P2 afterwards rearranges its schedule. After a series of communications with F2 and rearrangements of its schedule, P2 then gets available capacity from F2. P2 does the same things to F3 and finally gets an optimal schedule, as shown in Fig. 6. Fig. 6 shows that P2 will fulfill its DNPD project after 18 time units.

Fig. 5 P2’s Optimal Schedule (With Conflict)

Fig. 6 P2’s Optimal Schedule (Without Conflict)

P3, like P2, after iteratively communicating with F1, F2 and adjusting its self schedule, finally gets its schedule, as shown in Fig. 7. It takes 24 time units for P3 to fulfill its DNPD project.

Fig. 7 P3’s Optimal Schedule (Without Conflict)

Fig. 8 makes a summary of the aforementioned scheduling results, which shows that P1, P2 and P3 are expected to fulfill their projects by spending 16, 18 and 24 time units whereas F1, F2 and F3 complete their services by 20, 14 and 15 respectively.

Fig. 8 DNPD Schedules In The Design Chain Network

B. Improving DNPD Schedules Jointly

This subsection demonstrates how a core company coordinates with others in virtue of the satellite companies to improve its DNPD schedule and how the coordination contributes to the utilization rates of satellite companies.

The following demonstrates the process of how P2’s DNPD schedule is coordinated and improved:

1) P2 sends an adjustment request message to F2 to ask for adjusting the service time of f2 from 9–12 to 2–5.
2) F2, after receiving the request message from P2, checks the P2’s desired time period and finds that the 4–5 is reserved by P1. F2 then sends a further adjustment request message to P1, querying the possibility of putting off P1’s reserved service time to 5–7.
3) P1, receiving the request, will assess the effects of the adjustment to check whether the adjustment has any negative impact on its schedule. For P1, since f2 is not on the critical path of its schedule, the adjustment will not influence the completion time of its DNPD. That is, P1’s interest will not be harmed. P1 therefore replies with an acceptance message to P1.
4) F2, after receiving the acceptance from P1, immediately replies with an acceptance message to P2.
5) P2, after receiving the message from F2, sends another
adjustment request message to F3 to ask for the service time of f3 to be adjusted from 13~15 to 6~8.
6) F3, after receiving the request message from P2, checks its status of capacity loading, finds that the time period
7) P2, after receiving the acceptance message, sends adjustment messages to F2 and F3.
8) F2 and F3, receiving the adjustment messages, immediately adjusts its service time for f2 and f3 of P2.
   F2 further sends an adjustment message to P1.
   After going through the aforementioned process, P2’s schedule is improved, as shown in Fig. 9. Likewise, P3
   improves its schedule, as shown in Fig. 10. In this example, a design chain improvement brings an additional
   improvement opportunity to another design chain.
   After rescheduling for the design change as shown in the previous subsection, P1 can follow the coordination process
   for adjusting its NPD schedule.
   The coordination process, based on the Pareto improvement spirit, tells that design chains’ joint NPD
   schedules can be cooperatively improved without hurting anyone’s interests. Through the communication and
   coordination, P1, P2 and P3’s joint schedules are improved. Their schedule completion times are shortened to 16, 10 and
   23 respectively. Besides, F1, F2 and F3 will complete their services before 19, 13 and 13 as shown in Fig. 11. The
   improvements are summarized in Table I.

   Fig. 9 P2’s DNPD Schedule After Improvement

   Fig. 10 P3’s DNPD Schedule After Improvement

   Fig. 11 Improved Joint DNPD Schedules

<table>
<thead>
<tr>
<th>Organizations</th>
<th>Expected Completion Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P1</td>
</tr>
<tr>
<td>Before improvement</td>
<td>16</td>
</tr>
<tr>
<td>After improvement</td>
<td>16</td>
</tr>
</tbody>
</table>

C. Impact on Launch Time by Design Change

This subsection demonstrates the impact on the new product launch time made by a design change.

In this example, given that at time Point 2, P1’s new product specification is modified and the modification
extends p11’s required time from 3 to 6. P1 must reschedule its project to continue with its NPD plan.

To reschedule self DNPD project, P1 asks its outsourcing companies to adjust their previously reserved service times.
After iterative communications with F1, F2 and F3, a new schedule is generated, as shown in Fig. 12.

In Fig. 12, P1’s project will not be fulfilled until time point 32. That is, the project needs additional 13 time units.
The delay is expected to bring about profit loss. Moreover, for F1, F2 and F3, their service completion times are
expected to be deferred until 25, 15 and 29 time points respectively. Satellite companies’ capacity utilization rates
significantly fall.

D. Improving DNPD Schedules for Design Change

After rescheduling for the design change as shown in the previous subsection, P1 can follow the coordination process
described in subsection III.B to improve its DNPD schedules. The core companies and satellite companies finally get their new optimal schedules and capacity loading as shown in Fig. 13. The improvement effects can be seen in Table II.

The improvement shows that P1, P2 and P3’s expected project completion time are shortened from 32, 10 and 23 to 20, 10 and 21 respectively. P1’s schedule is improved and P3 also benefits from helping P1. Its new schedule is expected to be completed 2 time units earlier. Moreover, F1, F2 and F3’s service completion times are shortened from 25, 15, 29 to 17, 13, and 17 respectively.

The results show that the coordination can resolve arising conflicts caused by uncertainties and help organizations cooperatively to improve their joint DNPD schedules. Core companies in the distributed environment are likely to benefit from helping other companies. Such benefits will motivate organizations to coordinate their DNPD processes with each other.

IV. PROBLEM FORMULATION

The multiple design chains conflict problem illustrated in Section III is somewhat similar to the congestion game in which a core company corresponds to a game player and a satellite to a resource. The design chains and congestion game, nevertheless, are slightly different. First, in the design chains, resource cost is not taken into consideration. Second, in the design chains, players’ goals are not uniform, i.e., every design chain’ DNPD processes and projects are variant. Based on the above problem characteristics, we formulate the multiple design chains conflict problem.

The notations used for this problem are as follows:

- MDCP: multiple design chains coordination problem
- P: a set of core companies, \( P = \{P_1, \ldots, P_j, \ldots, P_J\} \)
- R: a set of resources provided by satellite companies, \( R = \{R_1, \ldots, R_k, \ldots, R_K\} \)
- \( \Sigma_j \): a set of \( P_j \)'s potential schedules to fulfil its DNPD processes \( \Pi_j \).
- \( \Pi_j \): a set of \( P_j \)'s DNPD processes to be fulfilled, \( \Pi_j = \{\Pi_{1j}, \ldots, \Pi_{ij}, \ldots, \Pi_{IJ}\} \).
- \( p_{ij} \): a set of activities which form the company \( P_j \)'s \( i^{th} \) DNPD process, \( p_{ij} = \{p_{f,ij}^1, \ldots, p_{f,ij}^m, \ldots, p_{f,ij}^M\} \).
- \( f_{ij} \): a set of outsourcing activities, \( f_{ij} = \{f_{ij}^1, \ldots, f_{ij}^m, \ldots, f_{ij}^M\} \).
- \( r(f_{mij}, R_k) \): a function which describes the units of \( R_k \)'s resources reserved for \( f_{mij} \) during time \( t \), \( r(f_{mij}, R_k, t) \) is a positive number.
- \( Hij \): a set of pairs \( (m, n) \) which describes the precedence relations between activities \( p_{f,ij}^m \) and \( p_{f,nij}^n \).
- \( \Pi_{ij} \): \( P_j \)'s \( i^{th} \) DNPD process.
- \( dm_{ij} \): the duration of activity \( p_{f,ij}^m \), \( dm_{ij} \) is constant in this study.
- \( \text{beg}_{mij} \): the begin time of activity \( p_{f,ij}^m \).
- \( \text{fin}_{mij} \): the finish time of activity \( p_{f,ij}^m \).
- \( \text{fin}_{Mij} \): the finish time of final activity \( p_{f,Mij}^M \).
- \( K_k \): \( R_k \)'s capacity during \( t \).
- \( \tau \): the \( \tau^{th} \) time period.

The problem is represented as shown in Equations (1) and (2):

\[
\text{MDCP} = \min_{P} \sum_{j=1}^{J} \text{Σ}_j > \text{subj. to: (1)} \]
\[
\Pi_i = \text{Σ}_j > \text{subj. to: (2)} \]

For a core company \( P_j \), its personal objective and constraints are expressed in Equations (3) to (7):

\[
\text{Minimize } \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{t=1}^{T} r(f_{mij}, R_k, t) \leq K_k \quad \text{(3)}
\]

Such that:
\[
\text{beg}_{mij} + dm_{ij} \leq \text{fin}_{mij} \quad \text{(4)}
\]
\[
\text{fin}_{ij} \quad \text{(5)}
\]

Besides, if all the organizations cooperatively optimize and improve their schedules, \( \Sigma_1, \ldots, \Sigma_j, \ldots, \Sigma_J \) will form joint optimal schedules for the multiple design chains.

V. DEVELOPMENT OF COORDINATION MECHANISM

An agent-based distributed coordination mechanism for multiple design chains is elaborated in this section. In the
coordination mechanism, core companies can first adopt local search methods and coordinate with satellite companies to individually arrange their optimal DNPD schedules subject to the constraints and then reserve satellite companies’ services accordingly. Afterwards, core companies can cooperatively improve their schedules through coordination with other companies. The cooperation follows the Pareto improvement rule; that is, any adjustment in the design chain must make at least one design chain better off without making anyone else worse off. The mechanism allows the involved companies to communicate with each other with only necessary information that ensures the privacies of the companies.

The coordination mechanism, including the agent platform, communication ontology for software agents, and the coordination process for resolving conflicts, is described as follows.

A. Creating Agents

The agent platforms for core and satellite companies are designed as shown in Figs. 14 and 15 respectively.

Fig. 14 shows the five physical agents for a core company, including querying agent, schedule maintaining agent, booking agent, requesting agent, and responding agent.

In Fig. 14, querying agent is designated to query about the possible earliest begin time of an outsourcing activity. Schedule maintaining agent is designated to manage the NPD schedule of a design chain. Its duties are: arranging self schedule, lock/unlock the to-be-adjusted activities, triggering requesting agent to send adjustment requests to satellite companies for improving self schedule, assessing the coming request to check whether the adjustment has negative impacts, commanding the responding agent to answer the coming requests, and adjusting self schedule. Booking agent is designated to make reservation with satellite companies according to the arrangement of schedule maintaining agent. Requesting agent is designated to follow the orders from schedule maintaining agent to send adjustment request messages to satellite companies. Responding agent, under the supervision of schedule maintaining agent, is designated to answer the coming requests from satellite companies.

B. Creating Agent Communication Ontology

All the interactions between core and satellite companies are held by message exchanges. Their messages are coded in a common communication language. The communication language is developed, based on the communication ontology shown in Fig. 16. Ontology is the specification of a conceptualization [42] which describes the concepts, concepts’ attributes and the relations between concepts. The ontology in this study contains 11 concepts, including CoreCompany, SatelliteCompany, Activity, FeasibleTimeQuery, FeasibleTimeResponse, Booking, Abandonment, AdjustmentRequest, Rejection, Acceptance, and Adjustment. The ontology in Fig. 16 will be implemented as a communication language to equip physical agents with communication abilities.
C. Creating Agent Behaviours for Coordination Process

The agents’ behaviours for the coordination process to resolve conflicts are divided into two parts: the coordination of an outsourcing activity and the coordination of a to-be-adjusted activity as shown in Figs. 17 and 18 respectively.

The sequence diagram in Fig. 17 shows the interaction between a core company \( P_j \) and its satellite company \( F_k \) for the coordination of an outsourcing activity. The steps of the sequence diagram and agents’ behaviors are described as follows:

1) \( P_j \)’s querying agent sends a \textit{FeasibleTimeQuery} message to \( F_k \) to query about the possible earliest begin time of the outsourcing activity (Equation (3)).
2) \( F_k \)’s capacity status managing agent checks the status of the capacity loading (Equation (7)) and calculates the earliest time to begin the activity.
3) \( F_k \)’s answering agent replies with the \textit{FeasibleTimeResponse} corresponding to the queried activity. \( F_k \)’s capacity status managing agent then temporarily holds the time period for the activity.
4) \( P_j \)’s schedule maintaining agent checks whether the replied time is acceptable (Equations (4), (5) and (6)).
5) \( F_k \)’s capacity status managing agent, once receiving the Booking message, makes a reservation for \( P_j \) and replies with the Booking message as a confirmation.
6) For \( P_j \), if the replied time is unacceptable, its schedule maintaining agent sends an Abandonment message to \( F_k \) or goes to Step 1 to re-query the same activity.
7) \( F_k \)’s capacity status managing agent, once receiving the Abandonment message, releases the held time.

Fig. 18 shows the process of coordinating a to-be-adjusted activity between core company \( P_j \) and its satellite company \( F_k \), and between \( F_k \) and other core companies \( P_j \). This coordination process can be initialized once \( P_j \) is not satisfied with its current DNPD schedule and has a better schedule to replace its current one. The steps of the sequence diagram and agents’ behaviors are described as follows:

1) \( P_j \)’s requesting agent sends an \textit{AdjustmentRequest} message to \( F_k \) to make a request for adjusting a specific activity’s execution time to a desired (better) one.
2) \( F_k \)’s capacity status managing agent, once receiving an AdjustmentRequest message, checks whether the adjustment will conflict with any other reserved service time (Equation (7)).
3) If the adjustment has any conflict over \( P_j \), \( F_k \)’s
coordinating agent calculates the conflicts, the conflicted P_j and new adjustments to resolve the conflicts. Afterwards, requesting agent sends further AdjustmentRequest messages to ask the conflicted P_j for adjusting their activity execution times.

4) P_j’s schedule maintaining agent, once receiving an AdjustmentRequest message, assesses the impact made by the adjustment.

5) If the assessment says that the adjustment has no negative influence, i.e., the beg_{j+1} in Equation (3) can be earlier or at least remain the same, the P_j’s schedule maintaining agent locks the activity and responding agent replies with an Acceptance message, otherwise, responding agent replies with a Rejection message.

6) F_k’s coordinating agent, once receiving all replies from P_j, checks whether all replies are Acceptance.

7) If all replies are Acceptance, F_k’s responding agent returns an Acceptance message to P_j.

8) P_j’s schedule maintaining agent, after receiving the Acceptance message, sends an Adjustment message to F_k.

9) F_k’s capacity status managing agent, once receiving an Adjustment, adjusts the execution time of the corresponding and other related activities, and then,

10) F_k’s coordinating agent informs P_j about the time adjustment of their activities.

11) If any one of the replies is Rejection, F_k’s responding agent delivers a Rejection message to P_j, and at the same time,

12) F_k’s coordinating agent asks P_j’s schedule maintaining agent to unlock their activities.

13) P_j’s schedule maintaining agent, once receiving Rejection the message from F_k, abandons the adjustment or go to Step 1 to try a new adjustment.

VI. EXPERIMENTS

Two experiments were carried out to observe the effectiveness of our proposed coordination mechanism in this section. The following subsections describe the experimental settings, the experimental design and evaluation, and the experimental results and discussions.

A. Experimental Settings

The experimental environment of the multiple design chains network is set up as follows:

1) Design chain networks: Ten design chains networks are generated for each experiment to be carried out ten rounds. In each network, 25 core companies along with ten satellite companies form 25 design chains, including one predetermined and 24 random design chains. Each design chain manages a DNPD project.

2) Pre-determined design chain: The pre-determined design chain is a simplified real world design chain that manages the critical DNPD activities for the development of an automatic special purpose machine. The DNPD process and its activities are described in Fig. 19 and Table III. The pre-determined design chain in this study is used to observe the impact if a design change happens and how the multiple design chains are autonomously coordinated afterwards.

3) Random design chains: The 24 random design chains are determined randomly for 10 networks respectively. Each of the random design chain in every network has six to ten activities. The predecessors of each activity, whether the activity is internal or outsourcing, and which satellite company can serve the activity are randomly determined. The duration of each activity is also randomly generated in the range of 5 to 13 time units.

4) Satellite companies: The ten satellite companies provide renewable and irreplaceable resources to the 25 core companies. Each satellite company can serve one core company during a time unit. The service costs of the satellite companies in this study are not taken into consideration.

<table>
<thead>
<tr>
<th>No</th>
<th>Activity Name</th>
<th>Predecessor</th>
<th>Satellite Company</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Spec development</td>
<td>-</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Concept sketching</td>
<td>1</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Detail drawing</td>
<td>2</td>
<td>-</td>
<td>7</td>
</tr>
</tbody>
</table>
Experimental Design and Evaluation

1) Experiment 1: Experiment 1 observes the improvement effects of our proposed mechanism. In this experiment, we use two approaches to make comparisons on the improvement effect of after and before coordination respectively. The first comparison uses our proposed distributed coordination mechanism (DCM) and the second a centralized approach (CA).

The DCM is applied in the distributed environment in which organizations autonomously coordinate with others if any improvement is desired. Since that in the distributed environment, no one knows complete information of the design chains network and moreover, a failed coordination might be successful later after some successful improvements, the appropriate time to terminate DCM will be hardly detected. To avoid the endless loop of coordination, we suppose that the DCM allows every core company at most ten continuous times failed in improving its DNPD schedule.

On the other, The CA is applied in an imaginary centralized environment in which all information of design chain organizations, including core and satellite companies, are centralized. We suppose that the CA schedules the DNPD processes of all design chains orderly according the time they enter the design chains network, and improves their schedules orderly and iteratively until no further improvement can be made.

2) Experiment 2: Experiment 2 observes the impact of design change and the improvement effects of our proposed distributed coordination mechanism on the DNPD schedules of multiple design chains network. In this experiment, a design change happens in the DNPD process of the predetermined design chain, in which the 3rd activity (detail drawing) extends its duration from 7 to 13. The design change leads the predetermined design chain to reschedule its DNPD process and then to improve its schedule. In this experiment, DCM is used to coordinate the conflict caused by the design change.

Every experiment was executed 10 rounds. To evaluate the experimental results, we use the Wilcoxon signed rank test (or Wilcoxon t-test), provided by IBM SPSS Statistics 19, to test the significance of the coordination effects. The null hypothesis of no improvement between after and before coordination in every experiment is tested.

Experimental Result and Discussion

1) Experiment 1:

Table IV shows the one tailed Wilcoxon signed rank test results of Experiment 1. The Wilcoxon signed rank test compares the DNPD process completion times of the design chains pairwisely between the after and before coordination for the application of DCM and CA respectively.

<table>
<thead>
<tr>
<th>Network#</th>
<th>z-score(p-value, 1-tailed)</th>
<th>z-score(p-value, 1-tailed)</th>
<th>z-score(p-value, 1-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>DCM</td>
<td>CA</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1.826(0.034*)</td>
<td>-1.826(0.034*)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-2.032(0.021*)</td>
<td>-2.226(0.013*)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1.633(0.051)</td>
<td>-1.00 (0.158)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-1.633(0.051)</td>
<td>-1.414(0.078)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-2.366(0.000**)</td>
<td>-2.524(0.006**)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-1.342(0.09)</td>
<td>-1.00 (0.158)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-1.826(0.034*)</td>
<td>-1.604(0.054)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-1.00 (0.158)</td>
<td>-1.00 (0.158)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>-1.826(0.034*)</td>
<td>-2.023(0.021*)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.000(1.000)</td>
<td>0.000(1.000)</td>
<td></td>
</tr>
</tbody>
</table>

*p<0.05,**p<0.01

In Table IV, by using DCM, the first, second, seventh and ninth design chains networks are improved in significant level and the fifth is improved in highly significant level. DCM in other four design chains networks also brings improvements to the DNPD schedules though the improvements are insignificant.

On the other, AC improves 4 out of 10 design chains networks significantly or highly significantly. The AC in other five design chains networks also carries out improvements to the DNPD schedules though the improvements are insignificant.

The tenth design chains network remain the same after applying the DCM and AC. It implies that for DCM and AC, it’s difficult to make improvement on this network.

From the above results, we get an implication that if the multiple design chains network is possible to make improvement on it, DCM performs as well as CA.
If we take a closer look into the statistical results in Table IV, we find that in the seventh designs chain network, DCM outperforms AC. The reason is that DCM takes advantage of software agents’autonomies to facilitate coordination and then DCM than AC selects design chain more randomly to make improvement. The randomness might be the contributing factor to the better improvement of complicated design chain network.

2) Experiment 2:

Table V shows the one tailed Wilcoxon signed rank test results of Experiment 2. The after and before coordination of using DCM to lessen the impact of design change are compared.

<table>
<thead>
<tr>
<th>DC Network#</th>
<th>z-score (p-value, 1-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2.803 (0.003**)</td>
</tr>
<tr>
<td>2</td>
<td>-1.826 (0.034*)</td>
</tr>
<tr>
<td>3</td>
<td>-2.20 (0.014*)</td>
</tr>
<tr>
<td>4</td>
<td>-2.64 (0.004**)</td>
</tr>
<tr>
<td>5</td>
<td>-1.50 (0.158)</td>
</tr>
<tr>
<td>6</td>
<td>-1.841 (0.035*)</td>
</tr>
<tr>
<td>7</td>
<td>-2.052 (0.021*)</td>
</tr>
<tr>
<td>8</td>
<td>-2.37 (0.009**)</td>
</tr>
<tr>
<td>9</td>
<td>-2.371 (0.009**)</td>
</tr>
<tr>
<td>10</td>
<td>-2.366 (0.009**)</td>
</tr>
</tbody>
</table>

In Table V, the statistical results show that DCM lessen the impacts of design changes and improves all the ten design chains networks. The statistical results further show that five out of ten networks are highly significantly improved and four are significantly improved.

Experiments 1 and 2 demonstrate the validity of the DCM and prove that the idea of resolving the conflicts illustrated in Section III is workable. Based on the Pareto improvement concept, the experimental results show that one improvement is likely to bring other improvements to the design chains network. Besides, some core companies benefit from helping other core companies improve their DNPD schedules. In the real environment, such benefits will motivate companies to cooperatively coordinate with others. Moreover, Satellite companies’ capacity utilizations in the experiments are also improved though their data are not shown here.

VII. CONCLUSION AND FUTURE RESEARCH DIRECTION

DNPD has an inevitable tendency: outsourcing non-core competence activities to other companies. The tendency forces the R&D-oriented companies to form distributed design chains network. In a distributed design chain network, resources provided by the satellite companies are finite and relations among the organizations are loosely-coupled and complicated. Under such a condition, any variation or uncertainty is likely to create conflicts that are difficult to resolve. These conflicts might delay the new product launch time and lead to serious profit loss.

To solve the conflict problem, we first introduce an example to demonstrate the occurring conflicts and how the conflicts could be resolved in distributed environment, formulate the problem as a congestion game, design an agent-based coordination method, implement the mechanism, and finally evaluate the mechanism. The experimental results show that this mechanism can help to lessen the impacts caused by the environmental variations or uncertainties, such as the design changes.

To advance this study, some future research directions are suggested. First, execute more scenario-based experiments to evaluate the performance of the coordination mechanism. Second, apply the coordination mechanism to larger scale problems or other problem domains. Third, reformulate the conflict problem as an optimization problem of both new product launch time and outsourcing cost, and last, enhance the efficiency and quality of improvement of the coordination mechanism.

ACKNOWLEDGMENT

The authors acknowledge support under NSC Grant NSC 99-2410-H-020-006-MY2.

REFERENCES


