INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

ACCELERATING THE PRODUCT DEVELOPMENT PROCESS IN A MULTI-PROJECT ENVIRONMENT USING DYNAMIC SEQUENCING METHOD

Chang Muk Kang and Yoo Suk Hong
Department of Industrial Engineering, Seoul National University

ABSTRACT

The design process is hard to accelerate due to its iterative nature which increases project cost and completion time, and is a major source of inefficient design processes. For this reason, much research intended to model iterations and to find the optimal structure of the process. However, these approaches have intrinsic limitations in that they can only be applicable to a single-project environment. In a multi-project environment, waiting time induced by resource shortage becomes more critical source of lengthy projects than iteration. In this paper, we propose dynamic sequencing method which reduces the waiting time of design process by changing the sequence of design tasks by sending waiting projects to an another idle resource. To evaluate the effect of this method, we developed a process model which is suitable for representing iterations and multi-project environment in an efficient manner and performed simulation using this model. The simulation results showed that the dynamic sequencing method is significantly better than traditional static sequencing method in terms of average duration of design projects. We also found that the improvement is more salient when projects were crowded and a design process was unbalanced which usually have negative effects on the process performance.

Keywords: Design process, Multi-project environment, Dynamic sequencing, DSM, Stochastic processing network

1 INTRODUCTION

Speed is one of the hottest issues in modern industries. The world-class companies emphasise the importance of speeding up their processes. Especially, the development speed plays a crucial role in success of a new product. To response rapidly changing market preference, companies endeavour to reduce their lead time of development by adapting many acceleration practices, such as concurrent engineering.

Product development project is composed of many tasks making discrete design decisions [1]. As the scale of a product increases, the number of decision makers grows exponentially and it becomes more complex to coordinate their decisions [2]. Development decisions are coupled with other decisions in the development of a complex product because one decision may require information from the result of other decisions. These information dependencies between design decisions become a cause of iteration of design tasks [1, 3-5]. Iteration is the nature of a design project and makes a design project more difficult to manage than any other routine projects. Ahmadi *et al.* [4] clearly noted that iteration increases project cost and completion time, and is a major source of inefficient processes.

Much research has endeavoured to model iteration for the purpose of obtaining the efficient structure of design tasks. Most of them were to find the optimal sequence of design tasks which guarantees the minimum iteration and project duration using analytic or simulation-based methods. Henceforth we call it a sequencing problem. While their efforts were successful to some extent, those approaches have their intrinsic limitations in that they can only be applicable to the management of a single project. In reality, however, multiple projects are concurrently under way and overloaded concurrent projects ruin the efficiency of development organisation [6].

This paper focuses on managing multiple design projects. Unlike the single-project environment, several projects compete for limited resources in a multi-project environment. The competition naturally induces waiting time which makes the project duration longer than it should be. In a multi-project environment, both waiting time and iteration time are major cause of lengthy development projects. Sometimes, waiting time is more critical than the iteration.

This paper proposes dynamic sequencing of design tasks as a means of reducing waiting time, which determines the task sequence dynamically according to the state of resources; a task which only consumes currently available resources regardless of a predefined task sequence. This sequencing method may reduce waiting time of individual projects while it cannot guarantee minimum iteration. It is only valuable when reduced waiting time exceeds additional iteration time. Simulation method is used to evaluate effect of the proposed method in such a trade-off. Among several metrics for measure project performance, the average duration (flow time) of multiple projects is used as a performance measure.

2 LITERATURE REVIEW

As mentioned above, iteration is the key factor in managing design projects. Iterations can be classified into two types: expected and unexpected iterations [3]. Expected iterations are ones which occur deterministically when the interrelated design tasks are processed in parallel. Many models for overlapping design tasks assume the expected iterations [7-9]. Smith and Eppinger [10] developed a work transformation matrix extending design structure matrix (DSM) to predict coupled tasks which will require many expected iterations.

On the other hand, the result of a previous design task in compatible with the current design task induces unexpected iterations in a sequential process. Therefore, the number of unexpected iterations and the duration of a design project depend on a sequence of design tasks. Existing research mainly focused on finding an optimal sequence of a single project which guarantees minimum number of iterations.

Smith and Eppinger [1] proposed the sequential iteration DSM which represents the probability of rework, as opposed to a binary metric, measured by the strength of dependency between two design tasks. They found the expected duration of a design project for a given process architecture in an analytic manner using a reward Markov chain. Ahmadi and Wang [11] also modelled the design process with a Markov chain. They intended to define the optimal acceptance level of design review in terms of the expected number of iterations. While these two studies assumed that the rework probability remains the same regardless of the number of iterations, Ahmadi *et al.* [4] modelled the rework probability to be discounted as the process iterates taking into account learning effect. The learning effect was classified into four types with respect to the change of rework probability by Andersson *et al.* [12]. In advance, Browning and Eppinger [13] took into account not only the rework probability but also the rework impact which represents the amount of rework for the purpose of assessing the risk of development project. They also allowed overlapping multiple tasks at the same time. Cho and Eppinger [14] built a richer process simulation model incorporating the learning effects, parallel iterations and resource constraints based on the work of Browning and Eppinger [13]. They proposed rework concurrency method which determines whether to execute parallel iterations and a heuristic method for allocating resource among competing tasks. From a different point of view, Clarkson *et al.*[15] proposed a method to visualize design processes and identify sequential and parallel tasks and proper sequence of tasks with respect to the confidence level of design parameters.

While most research dealt with managing a single design project, a design organisation handles multiple projects at the same time in practice. Furthermore, companies had a trouble with managing overloaded development projects as shown in [6]. Elonen and Artto [16] pointed out six problem areas in managing development projects. Among them, resource shortage and improper resource allocation is most salient in a multi-project environment.

The most prospective research on managing multiple development projects is Adler's work in [17]. Adler modelled a project as an entity and a resource as a station, as in the process model. It enables development projects to be managed with traditional process management techniques. Anavi-Isakow and Golany [18] proposed constant number of projects in process (CONPIP) and constant time in process (CONTIP), which control the number of projects and the duration respectively, extending the constant work-in-process (CONWIP) technique used in production process control. Cohen *et al.* [19] showed the way how to determine the optimal constant in a CONPIP environment using cross-entropy method. Narahari *et al.* [20] modelled iteration and waiting behaviour of multiple design and development projects in a combined manner with a queuing network and evaluated several acceleration techniques.

Although much research can be found in the areas of single project sequencing and multi-project control independently, it is hard to find research dealing with a sequencing problem in a multi-project environment. It should be noted that a sequence of design tasks is assumed to be a matter of decision, not a predefined constraint. The iterative nature of design decisions often makes it hard to say which one of the decisions should precede the other. In a multi-project environment, deviations from the optimal sequence may be beneficial because they might reduce the waiting time in the bottleneck queues at a cost of additional iterations. This paper used the simulation approach to evaluate and illustrate the effect of changing the sequence of a design project in a multi-project environment, and derive its managerial implications.

3 DESIGN PROCESS MODELING

We only considered a detail design stage out of the whole design process which includes concept design, detail design, testing, and so on. It should be noted that only the detail design process was modeled just for the sake of ease of analysis, and that the same analysis can be extended to other stages of design process without loss of generality.

Design project sequencing starts with modelling a design process. This section explains how rework is modelled using design structure matrix (DSM), which is the major part of design process modelling, and stochastic processing network model which enables to handle multiple projects with a single DSM model.

3.1 DSM-based rework modelling

Contrary to the traditional process models, design process model should be able to handle iterations caused by rework of the tasks in an efficient way. While there are several process modelling techniques, such as generalised evaluation and review technique (GERT), signal flow graph, and system dynamics, none of these models is suitable for representing the rework characteristics and exploring the alternative process architectures [13]. Some of the recent research utilise DSMs to cope with these shortcomings. The model we present in this paper is also based on the revised version of DSM to model rework of the tasks.

As mentioned above, rework is caused by information dependencies between design tasks. It has to be noted before describing our rework model that every information dependency is assumed to be soft dependency. Ahmadi *et al.* [4] divided information dependency relationships among design tasks into two different categories: soft and hard dependencies. While the order of two tasks liked with soft dependency can be exchanged, the hard dependency fixes their precedence relationship. Unlike other research which modelled the soft and hard dependencies in a separated manner, we developed a unified modelling framework in which the dependencies of both types can be presented on the same basis. In this framework, the hard dependencies are considered as a special case of soft dependencies. We will explain how it is possible after describing the soft-dependency case.

While Steward [21] stated how to use a DSM in managing design process, it can be utilised in various manners according to what the values in the matrix mean. Because the probability and impact of rework varies across the degree of dependencies between tasks, our process model represents the rework with DSM-based rework probability and impact matrices, $\boldsymbol{RP}$ and $\boldsymbol{RI}$, adopting the model of Browning and Eppinger [13]. Rework impact means relative amount of the rework to the original work. The meaning of rework probability and impact is, however, more close to that of Smith and Eppinger [1]. While subdiagonal and superdiagonal numbers in the matrices have different meanings with a predetermined sequence in the model of Browning and Eppinger [13], Smith and Eppinger [1] gave the same meaning to both of them regardless of sequence: the probability that a row task is reworked when a row task precedes column tasks and the result of a column task is incompatible with the previous result of a row task. We adopt the meaning of Smith and Eppinger [1] to interpret rework probability and its impact because we do not assume a concrete sequence, in other words, hard dependency, and deal with dynamic sequence change. In Figure 1, task A is reworked with probability of 0.5 when it is done before task B since the value $\boldsymbol{RP}_{AB}$ is 0.5. In the same manner, the amount of rework is 0.7 because the value of $\boldsymbol{RI}_{AB}$ is 0.7.

RP	A	B
A	-	0.5
B	0.8	-

RI	A	B
A	-	0.7
B	0.6	-

Figure 1. Rework probability and impact matrices

As mentioned above, we have no additional method to indicate hard dependency. It is treated as merely the special case of soft dependency. If two tasks *A* and *B* have hard dependency and *A* has to precede *B*, it is meaningless to do *B* before completing *A* because *B* has to be fully reworked after the completion of *A*. In this case, both rework probability and impact of *B* are 1. Therefore, hard dependency can be represented by rework probability and impact of 1 in soft dependency context. Then, adverse sequence of hardly dependent tasks would be automatically rejected while finding the proper sequences.

3.2 Stochastic processing network model

Most of the previous DSMs representing a design process are composed of tasks and their rework relationships while the detail is slightly different for researchers. They are sufficient for representing a single-project environment because all tasks belong to the same project. On the other hand, tasks in a multi-project environment are owned by different projects and their ownership has to be modelled in the process model. This situation can hardly be represented by traditional models focusing on tasks.
As an alternative to these models, our process model adopted the idea of Adler *et al.* [17]. They proposed to model the product process as a stochastic processing network which is a collection of workstations through which a project passes to perform its constituent tasks in a process management perspective. Each workstation is composed of identical resources or employees who perform the specified tasks and the path that a project follows, i.e. the sequence of tasks, is stochastically determined in this model. That is, a design process is modelled as an open queuing network. While the DSMs in our process model still represent the rework of individual tasks, each project owns different DSMs and follows the network according to its own rework characteristics. This model facilitates the process modelling of multiple design projects because individual projects can be easily modelled as entities flowing through the network.
In addition, if we assume that parallel tasks are not allowed, i.e. all tasks are performed sequentially, the network is simplified into an open tandem queuing network as Figure 2. The transition probabilities between queues are specified by the DSMs we have already defined. This simplified model is sufficient for achieving the intended purpose of the analysis because this paper only concerns the sequence of design tasks. It is beyond the scope of this paper to analyse the effect of parallel processing. It is another research issue we plan to uncover in the future.

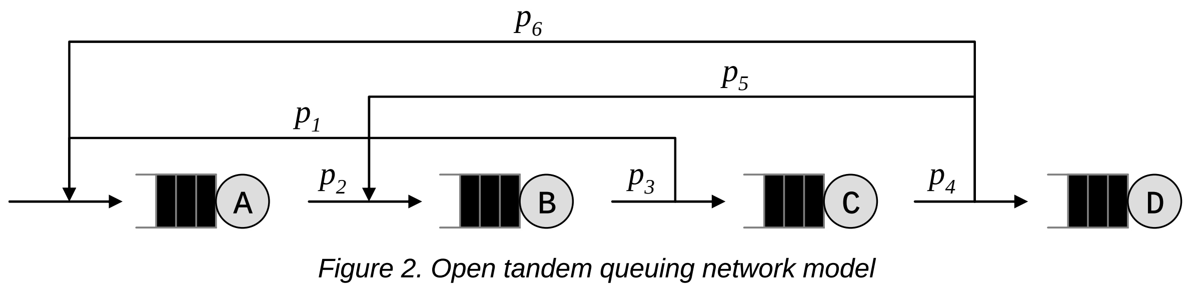


Figure 2. Open tandem queuing network model

3.3 Mechanism updating rework characteristics

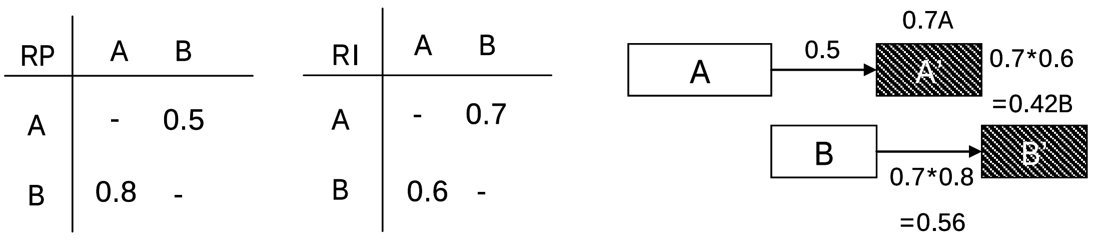
Another issue in design process modelling is how many times of iteration are allowed and how to reflect decrease in rework probability and impact. Most of research allowed infinite iterations though its probability converged to zero. While some research assumed iteration probability remains constant regardless of the number of iterations, it is more common in practice that probability and amount of rework decreases as rework repeats. It is one of the alternatives to incorporate learning effect to reflect this nature [13, 14]. However, it is hard to measure learning effect quantitatively and objectively and it is not certain whether learning effect is on rework probability, impact, or both.

To remedy this problem, we adopt the mechanism which determines amount of rework induced by overlapping tasks in model-based overlapping literatures. When overlapping coupled tasks, some of the downstream work during overlap may be obsolete if the result of unforeseen upstream task differs from the expected [8]. Such obsolete work has to be reworked and amount of the rework is the increasing function of overlapped work.

While sequential iteration is not induced by overlapping, sequential iteration also occurs when the unforeseen result of downstream task is incompatible with the upstream task. Then, we can assume that the probability and impact of rework are increasing functions of amount of unforeseen coupled work, henceforth referred as to $g(r_i)$.

In Figure 3, since $\boldsymbol{RP}_{AB}$ is 0.5 and $\boldsymbol{RI}_{AB}$, task *A* is reworked with the probability of 0.5 and the amount of rework is 0.7 of original work after task *B* is completed. If task *A* is reworked, task *B* consequently precedes rework of task *A*, denoted by *A'*. When task *B* was completed, the result of rework *A'*, 0.7 of the original work, was unforeseen and this also is the cause of reworking task *B*. If task *B* initially precedes original work of task *A*, 0.6 of task *B* is reworked with the probability of 0.8 according to ***RP*** and ***RI***. Then, it is rational conclusion that the probability and impact of rework of task *B* which precedes rework of task *A'* is $0.8\times g(0.7)$ and $0.6\times g(0.7)$ respectably. If uncertainty is evenly distributed throughout the work, $g(r_i)=r_i$ and rework probability and impact are $0.56(=0.7\times0.8)$ and $0.42(=0.7\times0.6)$ as Figure 3.

In this senses, rework probability and impact matrices, ***RP*** and ***RI***, should be dynamically updated whenever rework occurs. When impact of the current rework of task *i* is r_i, $\boldsymbol{RP}_{xi}$ and $\boldsymbol{RI}_{xi}$ should be updated to $r_i\boldsymbol{RP}^0_{xi}$ and $r_i\boldsymbol{RI}^0_{xi}$ for all tasks *x* which are completed already, where $\boldsymbol{RP}^0$ and $\boldsymbol{RI}^0$ are initial rework matrices.



RP	A	B
A	-	0.5
B	0.8	-

RI	A	B
A	-	0.7
B	0.6	-

Figure 3. Iteration mechanism

There is another problem between more than two coupled tasks. Assume that two tasks *A* and *B* precede task *C* and result of task *C* requires rework of task *A* and *B* as Figure 4. If rework of task *A* precedes rework of task *B*, the amount of rework of task *B* is hard to predict because rework impact from task *C* is 0.5 and that from task *A* is 0.3. In our model, we choose the maximum of them and it is a reasonable assumption in the practical perspective.

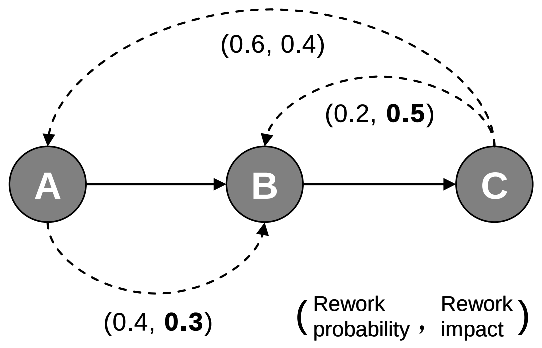


Figure 4. Three coupled tasks

While our updating mechanism reflects more rational assumptions about iterations, it is hard to analyse the model with this mechanism in an analytic manner. Instead, we used simulation model to implements the complex and dynamic rework-updating mechanism.

4 DYNAMIC SEQUENCING IN MULTI-PROJECT ENVIRONMENT

We consider managing multiple design projects in the perspective of minimising the duration (flow time) of a project. In a multi-project environment, more than two projects compete for a resource even if all tasks in single project are executed by independent resources. It results in resource shortage and duration of a design project lengthens. As more projects are loaded on the design organisation, waiting time in process dominates processing time.

A closer look reveals that most of waiting time occurs in front of the bottleneck resources. If a project does not wait for a bottleneck resource and works on another idle task first, it consequently reduces waiting time. In traditional production processes, it is impossible to change the task sequence because each task requires physical change of material. On the other hand, design process deals with information. Moreover, sequence is merely a decision variable in processing coupled tasks. Therefore, changing sequence according to the state of resources, henceforth we call it dynamic sequencing, may significantly reduces duration of a design project by reducing waiting time at the bottleneck.

There might be several available methods in dynamic sequencing. A novel algorithm or heuristic which improves performance of dynamic sequencing can be developed. However, it is out of the scope of this paper, and an issue of the further research. Instead, we implement a dynamic sequencing method which sending a project to an idle workstation if the next workstation in the optimal sequence is busy already.

While sequence change is available in design processes, following non-optimal sequence incurs additional iterations. They do not only lengthen the duration of a project, but also consumes the capacity of the design organisation. For this reason, trade-off between reduced waiting time by dynamic sequencing and excessive processing time by additional iteration has to be evaluated. We use simulation method to do this.

5 SIMULATION STUDY

Although Narahari *et al.* [20] already developed analytic queuing network model to estimate the duration of multiple design projects, it cannot be applied to develop analytic model in our context for two reasons. First, rework probability and impact is dynamically updated whenever a task is done. Second, it depends on the states of workstations which task is to be processed next when dynamic sequencing is used. Due to these properties, the routing is non-Markovian and it is hard to develop an analytic model. That is why we use simulation model to evaluate the effect of dynamic sequencing.

5.1 Case description

In this paper, optical mouse design project is presented as a sample case. All projects are assumed to be identical, which means that their task durations, process structure, and resource consumption are identical. The process consists of nine design tasks each of which designs a part of the optical mouse. The list of the parts and expected duration of designing each part is presented in Table 1. Each task is executed by a unique workstation; there is one-to-one matching relationship between tasks and workstations. To reflect uncertainty in processing tasks, we assume 20% of deviation from the expected duration using triangle distribution.

Table 1. Task durations

ID	Design task	Exp. duration
1	Connection cable design	2
2	Top case design	10
3	Bottom case design	8
4	Wheel mechanism design	6
5	Main board design	3
6	Microprocessor design	9
7	Click mechanism design	7
8	Balancing weight design	2
9	Optical sensor design	7

Figure 5 shows iteration structure. As described above, rework probability is the probability that a row task is reworked when a column task is done after the completion of the row task. Rework impact is amount of the corresponding rework.

Rework probability matrix (***RP***)

		1	2	3	4	5	6	7	8	9
Connection cable	1	■	1	1			1			
Top case	2		■	0.1	0.5	0.4				
Bottom case	3		0.1	■			0.1			0.8
Wheel mechanism	4		1.0	0.2	■	0.8			0.1	
Main board	5				0.2	■	0.8			
Microprocessor	6	0.1		0.7		0.3	■	1.0		1.0
Click mechanism	7						0.1	■		
Balance weight	8		0.1						■	
Optical sensor	9			0.1			0.1			■

Rework impact matrix (***RI***)

		1	2	3	4	5	6	7	8	9
Connection cable	1	■	0.5	0.5			0.5			
Top case	2		■	0.1	0.5	0.3				
Bottom case	3		0.1	■			0.8			0.5
Wheel mechanism	4		0.8	0.2	■	0.2			0.1	
Main board	5				0.4	■	0.1			
Microprocessor	6	0.2		0.9		0.1	■	0.8		0.5
Click mechanism	7						0.1	■		
Balance weight	8		0.1						■	
Optical sensor	9			0.5			0.5			■

Figure 5. Rework probability matrix (RP) and rework impact matrix (RI) for optical mouse design process

The task durations and the iteration structure may differ from the actual cases of mice manufacturers. Nevertheless, simulation with this virtual case is enough to provide meaningful results because the simulation model do not requires case-specific input information, but general information which can be obtained from any products. Cho and Eppinger [14] also utilized mostly the same information for developing a simulation model of uninhabited aerial vehicle design process.

We first find the optimal sequence which shows minimum average duration for a single project and the worst sequence which shows maximum average duration. We evaluated effect of dynamic sequencing comparing to these sequences.

5.2 Optimal sequence and worst sequence of a single-project

We intend to compare the performance of static sequencing and dynamic sequencing in a perspective of project duration. For fair comparison, projects with static sequencing should follow the optimal sequence. The optimal sequence is not the best practice utilized in industries, but the sequence that minimizes iteration time in a design process. Practices in real companies may be different from the optimal sequence due to a series of reasons such as availability of resources. This situation is in line with the purpose of this study which is to show that the minimum iteration does not guarantee the best performance in a multi-project environment unlike a single-project environment.

It clear that travelling salesman problem (TSP) reduces to task sequencing problem even iteration and duration are deterministic, that is, it is a NP-Hard problem. Because no exact algorithm is known to solve this kind of problems, we utilise genetic algorithm (GA) which is one of the meta-heuristic algorithm to find a near optimal sequence. Rogers *et al.* [22] already used GA to optimise the ordering of tasks with deterministic iteration. Table 2 shows the parameters of genetic algorithm we used. Due to the stochastic properties of our process model, we repeated 10,000 times of simulation for each sequence.

Table 2. GA parameters

Parameter	value
Number of populations	10
Population size	100
Reproduction ratio	10%
Crossover ratio	89%
Mutation ratio	1%
Crossover method	PMX method[a]
Stopping criterion	$\|Improvement\| < 0.2$

[a]From [23]

Using GA, we obtained the optimal sequence and the worst sequence. Expected duration and standard deviation for these sequences are presented in Table 3. The worst sequence shows about two times longer duration than the optimal sequence. In rework probability and impact matrices, numbers in a row generally indicate that the corresponding task incurs much iteration if it is performed before other tasks. On the other hand, numbers in a column indicate that the corresponding task should be performed earlier to minimise iterations. Task 1 which has three high values in its row and only one low value in its column locates the last position in the optimal sequence while it takes the first position in the worst sequence.

Table 3. Optimal and worst sequences

Optimal sequence	5, 9, 3, 2, 7, 6, 4, 8, 1
Optimal exp. duration	59.074
Standard deviation	4.478
Worst sequence	1, 6, 8, 3, 9, 7, 4, 5, 2
Worst exp. duration	108.099
Standard deviation	13.205

5.3 Static sequencing vs. dynamic sequencing

We first illustrate the performance of static sequencing and dynamic sequencing. In static sequencing, all projects follow the optimal sequence of single project and no sequence change is allowed. On the other hand, dynamic sequencing allows a project to go to another workstation if next workstation in the optimal sequence is busy.

These two sequencing methods show statistically significant difference in duration and waiting time. Figure 6 shows the average duration and standard deviation of durations of 20,000 projects with respect to average interarrival time. When interarrival time is 13, average duration decreased by 37.1% and standard deviation decreased by 44.5% with dynamic sequencing than with static sequencing. The result shows that dynamic sequencing makes the design process more efficient and robust.

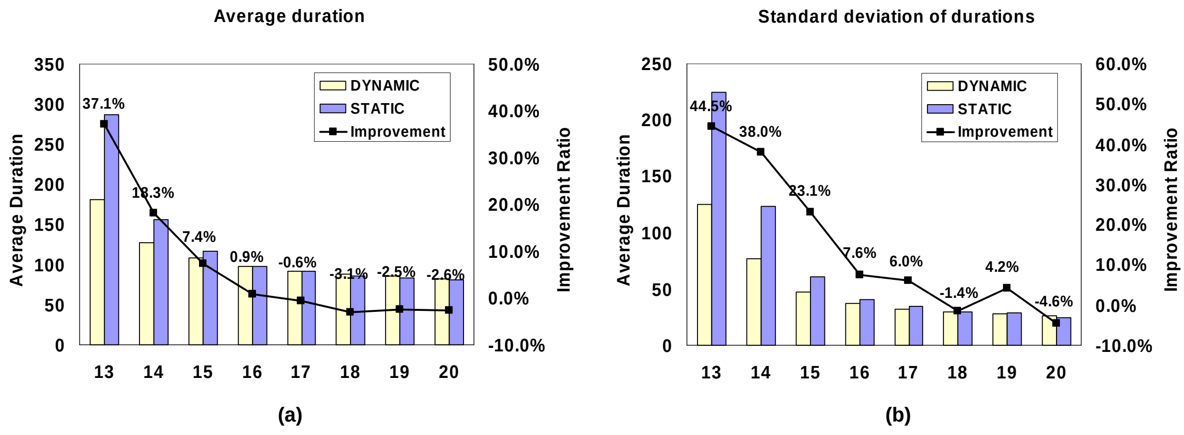


Figure 6. (a)Average duration and (b)standard deviation for each sequencing method with respect to interarrival times

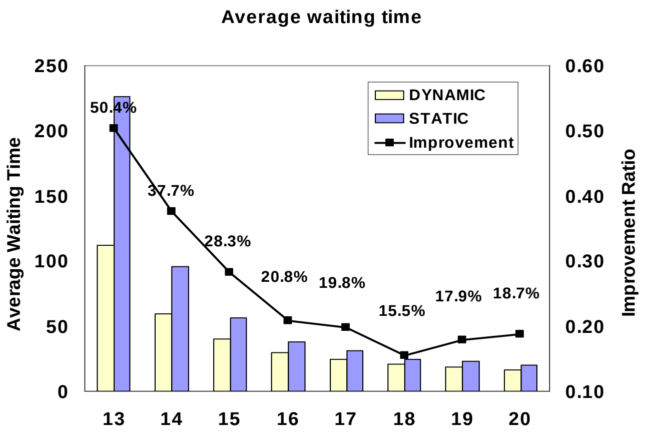


Figure 7. Waiting time between for each sequencing method with respect to interarrival times

The improvement by dynamic sequencing can be explained by elimination of waiting time. Under the dynamic sequencing rule, a project doesn't wait a preceding project to be finished and find an available workstation which can be executed immediately. It results much shorter waiting time as Figure 7.

Another finding is that the improvement lessens as the average interarrival time is getting longer. The difference of average durations disappears when average interarrival time is 16 units, and dynamic sequencing shows worse performance than static sequencing for the longer average interarrival times. The waiting time decreases as the interarrival time gets longer. Then, reduced waiting time by dynamic sequencing is not bigger than increased processing time by additional iterations. Moreover, the chance of waiting preceding projects to be finished decreases and dynamic sequencing is rarely needed. Therefore, dynamic sequencing is more effective as interarrival times decrease.

Dynamic sequencing method distributes the load of work uniformly to the all workstations. To illustrate it more explicitly, we draw utilisation and average number in queue graph with interarrival time 13. According to our simulation setting, the workstation 2 is a bottleneck and most of waiting time occurs in front of it since task 2 takes longest time, 10. Figure 8(a) shows that utilisation is more evenly distributed to all workstations with dynamic sequencing. As Figure 8(b), average number in queue at workstation 2 decreased to 6.6 from 16.0. We can see that workstation 3, 4 and specially 6 share waiting projects at workstation 2.

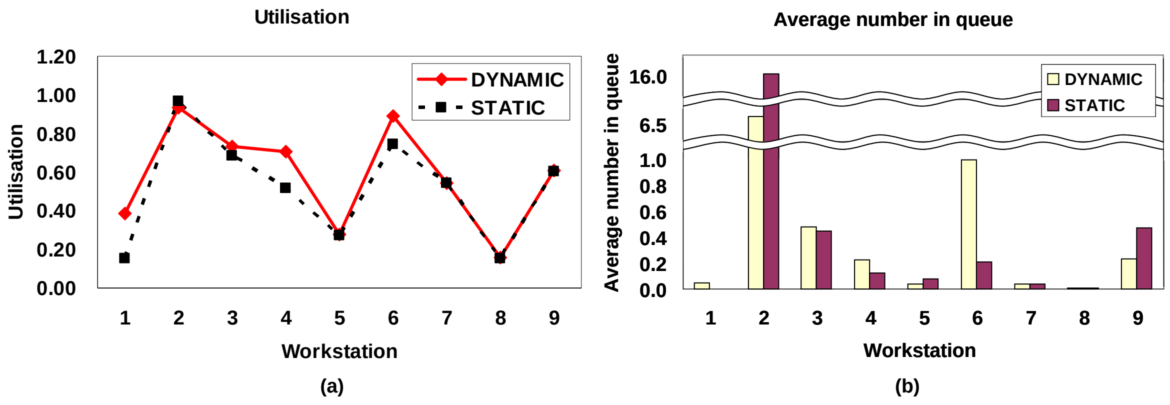


Figure 8. (a)Utilisation and (b) average number in queue for each sequencing method

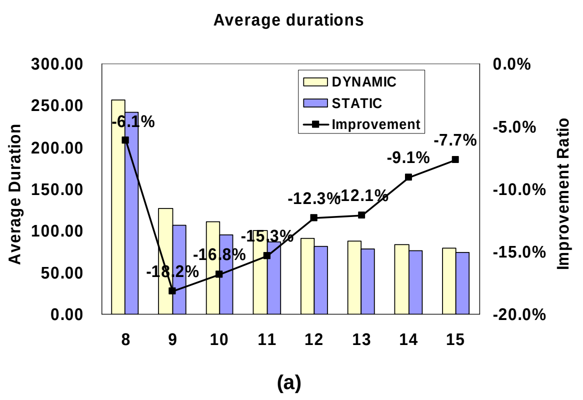
Additionally, static sequencing with the worst sequence shows much worse performance than dynamic sequencing. The system explodes when average interarrival time is shorter than 36 with the sequencing method. It is evidence that iteration has pretty much impact on duration of a design project in a multi-project environment.

5.4 Balanced process vs. unbalanced process

Dynamic sequencing improves design process by reducing waiting time at the bottleneck. In this sense, the effect of dynamic sequencing may vary according to how process is balanced. To validate this assumption, we design a totally balanced design process. The process consists of the same tasks as the original optical mouse design process except their durations are same with 6. The total durations of both the original unbalanced process and the balanced process are 54. The optimal sequence of the balanced process is (8, 7, 5, 9, 2, 3, 4, 6, 1).

Figure 9 shows durations and waiting times of the balanced process for each sequencing method. Contrary to the case of the unbalanced process, dynamic sequencing shows inferior performance to static sequencing for any interarrival times. Moreover, waiting time is longer with the dynamic sequencing.

The reason that dynamic sequencing shows worse performance can be explained by utilisation of individual workstations as Figure 10. Although utilisation of workstation 2 is a little low, other workstations, especially workstation 1, show much higher utilisation. It means that the design organisation work more with dynamic sequencing than with static sequencing.



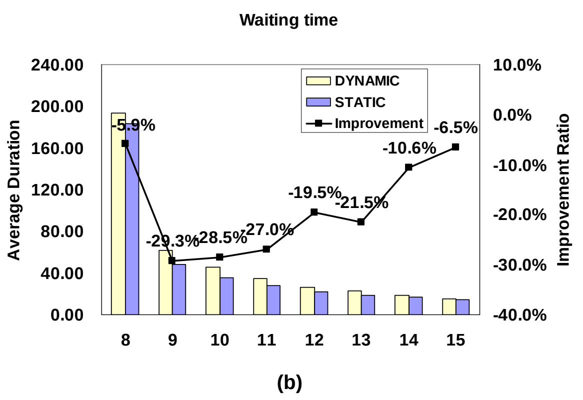


Figure 9. (a)Average durations and (b)waiting times of the balanced process

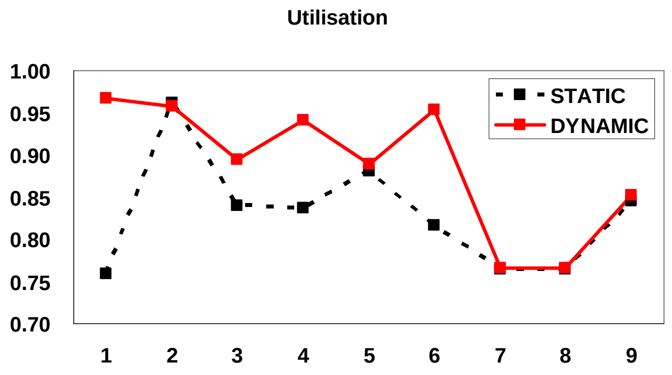


Figure 10. Utilisation of balanced process

Deviations from the optimal sequence make more iteration in processing a design project. It is also the case for an unbalanced process. However, dynamic sequencing of a balanced process cannot come over the additional processing time by reducing waiting time. There are three reasons for this. First, there is no bottleneck which incurs significantly long waiting time. Hence, the amount of reduced waiting time decreases for a balanced process. Second, highly iterative tasks take larger portion in total processing time. It means that one time rework has more impact on the whole project duration. The last reason is that there is few chances to send a project to an idle workstation since states of most of workstations would be same with busy or idle.

6 DISCUSSIONS

The result of the simulation gives us implications for design process management and our dynamic sequencing method.

6.1 Dynamic sequencing is effective in a multi-project environment

According to Figure 6 and 8, waiting time takes most of processing time both with static sequencing and dynamic sequencing when design projects are relatively crowded. Specially for average interarrival time 13, reduced waiting time by dynamic sequencing is 60, which comes over additional processing time even it is twice the original . Though it is not presented in the graphs, the system explodes much slowly with the dynamic sequencing when interarrival time is shorter than 13. These results tell us that dynamic sequencing is beneficial enough to accept additional iterations in a multi-project environment.

6.2 Sequencing is more significant in a multi-project environment

In Table 3, the worst sequence tasks about twice time than the optimal sequence. On the other hand, it is almost impossible to manage multiple design projects with the static sequencing using the worst sequence; the system explodes with shorter interarrival times than 36 which is about 3.5 times longer than the bottleneck duration, 10.

It means that a task sequence has much more impact on efficiency of a design organisation in a multi-project environment than in a single-project environment. An improper sequence incurs many unnecessary iterations. In a single-project environment, iterations merely lengthen the processing time of a project. In a multi-project environment, however, iterations consume additional resources and it makes longer waiting time of other projects. The impact of iteration propagates to many other projects and it results in explosion of the system.
It is a trivial result that dynamic sequencing based on the optimal sequence and based on the worst sequence shows only a little difference in performance: the system explodes shorter than 15 average interarrival times based on the worst sequence while it does shorter than 13 based on the optimal sequence. It is interpreted as dynamic sequencing is robust to a predefined sequence. In fact, it is hard to find even a near optimal sequence because optimising sequence problem is a NP-Hard problem as mentioned above. Moreover, there are many causes to follow a predefined sequence in real world design projects. Therefore, dynamic sequencing which is immune to failure of finding a proper sequence is more pragmatic method to manage multiple design projects.

6.3 Dynamic sequencing is an efficient balancing method

Balancing a process is one of the major techniques to improve efficiency of a process with static sequencing. However, it is a costly balancing method to add additional resources to s bottleneck. Dynamic sequencing is a kind of balancing methods. Idle resources share the load of a bottleneck resource by sequence change. It is an efficient balancing method because it does not require additional resources or flexible resources which can perform multiple tasks. As the process is more unbalanced, it gets more benefit from dynamic sequencing.

6.4 Iteration is the most important factor in developing a dynamic sequencing algorithm

To improve the effect of dynamic sequencing, a more efficient sequencing logic is needed. Most important factor in developing a dynamic sequencing algorithm is induced iterations by sequence change. In principle, a project should not change the optimal sequence if waiting time is shorter than the time needed by additional iterations. If the sequence is changed, which task induces least iterations should be selected.
An analytic model to predict iterations induced by sequence change is needed to develop such a sequencing algorithm. While it may be an exact model or an approximate model, complexity of the prediction model should be polynomial to solve the sequencing problem in reasonable time. Since simulation approach used in this paper takes pretty much time to predict duration or iteration, it is not applicable to explore sequencing alternatives.
Development of iteration prediction model and sequencing algorithm is our future research. We might get a hint of modelling and developing algorithms from dynamic routing literatures. While dynamic sequencing in a multi-project environment has never been researched, dynamic routing in queuing network has been researched in a little different context: choose a most efficient route among multiple parallel routes in a network. Ephremides *et al.* [24] studied performance of three dynamic routing policies in queuing network: Send-to-Shorter Queue (SS), Round-Robin (RR), and Send-to-Expected Shorter Queue (SES). Meanwhile, Kelly and Laws [25] modelled open queuing networks in the context of dynamic routing and job selection.

7 CONCLUSIONS

Iteration is a major cause of delay and excessive cost of design projects. Much existing research intended to minimise iteration and project duration by restructuring design process. They modelled a design process using Markov chain or simulation approach. Their effort was successful to optimise the design process in a single-project environment.
We focused on multi-project environment. While tasks in a project do not compete for a constrained resource, multiple projects have to compete for the resource since it cannot be shared by them. In this situation, waiting time cause by resource shortage might be more critical source of lengthy projects than iteration in a multi-project environment. Therefore, we suggest dynamic sequencing method which reduces waiting time by changing task sequences according to the states of resources. We used simulation approach to evaluate effect of the dynamic sequencing.

First, we modelled design process different from existing process models. We assume that tasks in a design project perform sequentially on a unique workstation. To model iterations, DSM-based rework probability and impact matrices are adopted from Browning and Eppinger [13]. The most distinct point in our model is rework-updating mechanism. Contrary to existing research, our model updates rework probability and impact matrices according to the amount of current rework which is determined by rework impact. It is a more rational modelling approach than updating only rework probability with learning coefficients. Using this model, we evaluate the effect of dynamic sequencing by simulation approach.

We compared performances of static sequencing with the optimal sequence and dynamic sequencing with respect to durations of design projects. Dynamic sequencing showed better performance than static sequencing in terms of average duration of multiple projects. The effect was more salient when projects are crowded and the process is unbalanced. Dynamic sequencing can be considered as a kind of balancing method which enables bottleneck resources to share their load with other resources by doing other tasks first.

The result also showed that a sequence is much more important in a multi-project environment than in a single-project environment. While improper sequencing in a single project merely incurs additional processing time, it also induces resources shortage in a multi-project environment. Dynamic sequencing does not only reduces waiting time, but also protect design projects from the improper sequencing.

This paper is a preliminary research to show that dynamic sequencing is a viable method for reducing the development durations in a multi-project environment. In the future searches, an analytic model for predicting process time and an efficient dynamic sequencing algorithm which improves the effect of dynamic sequencing should be developed. In addition, it is another research direction to incorporate other performance measures such as tardiness, cost, and resource utilisation.

REFERENCES

[1] Smith, R.P. and Eppinger, S.D. A predictive model of sequential iteration in engineering design. *Management Science*, 1997, 43(8), 1104-1120.

[2] Ulich, K.T. and Eppinger, S.D. *Product design and development*, 2003 (McGraw-Hill, New York).

[3] Smith, R.P. and Eppinger, S.D. Characteristics and models of iteration in engineering design. In *International Conference on Engineering Design*, The Hague, August 1993.

[4] Ahmadi, R., Roemer, T.A. and Wang, R.H. Structuring product development processes. *European Journal of Operational Research*, 2001, 130(3), 539-558.

[5] Eppinger, S.D., Whitney, D.E., Smith, R.P. and Gebala, D.A. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 1994, 6(1), 1-13.

[6] Wheelwright, S.C. and Clark, K.B. Creating project plans to focus product development. *Harvard Business Review*, 1992, 70(2), 70-82.

[7] Krishnan, V., Eppinger, S.D. and Whitney, D.E. A model-based framework to overlap product development activities. *Management Science*, 1997, 43(4), 437-451.

[8] Roemer, T.A. and Reza, A. Concurrent crashing and overlapping in product development. *Operations Research*, 2004, 52((4)), 606–622.

[9] Roemer, T.A., Reza, A. and Wang, R.H. Time-cost trade-offs in overlapped product development. *Operations Research*, 2000, 48(6), 858-865.

[10] Smith, R.P. and Eppinger, S.D. Identifying controlling features of engineering design iteration. *Management Science*, 1997, 43(3), 276-293.

[11] Ahmadi, R. and Wang, R.H. Managing development risk in product design processes. *Operations Research*, 1999, 47(2), 235-246.

[12] Andersson, J., Phol, J. and Eppinger, S.D. A design process modeling approach incorporating nonlinear elements. In *ASME International Design Engineering and Technical Conferences*, Atlanta, September 1998.

[13] Browning, T.R. and Eppinger, S.D. Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management*, 2002, 49(4), 428-442.

[14] Cho, S.-H. and Eppinger, S.D. A simulation-based process model for managing complex design projects. *IEEE Transactions on Engineering Management*, 2006, 52(3), 316-328.

[15] Clarkson, P.J., Melo, A. and Eckert, C., Visualization of routes in design process planning, In *IEEE International Conference on Information Visualization*, Los Alamitos, July 2000.

[16] Elonen, S. and Artto, K.A. Problems in managing internal development projects in multi-project environments. *International Journal of Project Management*, 2003, 21(6), 395-402.

[17] Adler, P.S., Mandelbaum, A., Nguyen, V. and Schwerer, E. From project to process management: An empirically-based framework for analyzing product development time. *Management Science*, 1995, 41(3), 458-484.

[18] Anavi-Isakow, S. and Golany, B. Managing multi-project environments through constant work-in-process. *International Journal of Project Management*, 2003, 21(1), 9-18.

[19] Cohen, I., Golany, B. and Shtub, A. Managing stochastic, finite capacity, multi-project systems through the cross-entropy methodology. *Annals of Operations Research*, 2005, 134(1), 183-199.

[20] Narahari, Y., Viswanadham, N. and Kumar, V.K. Lead time modeling and acceleration of product design and development. *IEEE Transactions on robotics and automation*, 1999, 15(5), 882-896.

[21] Steward, D.V. The design structure system: A method for managing design of complex systems. *IEEE Transactions on Engineering Management*, 1981, 28(3), 71-74.

[22] Rogers, J.L., McCully, C.M. and Bloebaum, C.L. Integrating a genetic algorithm into a knowledge-based system for ordering complex design processes. In *Artificial Intelligence in Design Conference*, Stanford, June 1996.

[23] Murty, K.G. *Operations research: Deterministic optimization models*, 1995 (Prentice Hall, Englewood Cliffs, NJ).

[24] Ephremides, A., Varaiya, P. and Walrand, J. A simple dynamic routing problem. *IEEE Transactions on Automatic Control*, 1980, 25(4), 690-693.

[25] Kelly, F.P. and Laws, C.N. Dynamic routing in open queueing networks: Brownian models. *Queueing Systems*, 1993, 13(1), 47-86.

Contact: Yoo Suk Hong
Seoul National University
Industrial Engineering
Shillim-dong, Kwanak-gu
Seoul
South Korea
+82-2-880-9070
+82-2-889-8560
yhong@snu.ac.kr
http://product.snu.ac.kr.