Proceedings of the 2011 Industrial Engineering Research Conference
T. Doolen and E. Van Aken, eds.

Improving Product Development through Front-Loading and Enhanced Iteration Management

Cecilia Martínez and Jennifer A. Farris
Department of Industrial Engineering
Texas Tech University, Lubbock, TX 79409, USA

Geert Letens
Department of Economics, Management and Leadership
Royal Military Academy, 1000 Brussels, Belgium

Abstract

Iteration is a central issue in the management of product development (PD) projects. Iteration is often recognized as a major source of increased PD lead-time and cost, a key driver of schedule risk, and a source of major uncertainties in the commitment of resources. However, iteration, when planned and managed effectively, can overcome the uncertainties inherent in interdependent development activities and thus, improve and accelerate PD projects. Based on case study insights, we argue that effective iteration management not only entails the elimination of unnecessary iterative loops due to ill-defined process structures but also a combination of iteration front-loading and iteration end-front separation. As such, this work complements the front-loading principles from the Lean PD literature by providing an alternative approach based upon improved iteration management. First, we use the design structure matrix to identify sets of iterative loops and then, we apply a binary ordering algorithm to front-load iterations within each loop set. Finally, we evaluate the process structure performance to quantify the iteration impact on the overall project completion time and identify the optimal PD process structure. The paper concludes with avenues for future research.

Keywords
Iteration management, front-loading, product development process, project management, DSM.

1. Introduction and Background
Reducing product development (PD) lead time—an important measure of PD process performance—continues to be a strategic priority for many organizations due to increased global competition. Effective planning is considered to be a critical factor in shortening the duration of PD projects. In a recent study of Japanese PD projects, Verworn et al. [1] observed that intensive planning at the fuzzy front end of the project directly affects not only the project efficiency but also the effectiveness, by reducing market and technical uncertainties. Similarly, Turtle (as cited by [2]) emphasized that good planning may result in fast PD projects, while poor planning may cause up to 70% of project delays. However, the nature of PD processes is iterative and uncertain, which makes managing PD projects a problematic and difficult endeavor, especially when traditional management techniques do not consider iteration. Consequently, the acceleration and improvement of the PD process, through planning, demands a deep understanding of design iterations [3].

Iteration is a fundamental and unavoidable characteristic of PD processes [4,5]. Through iterations, design problems are solved—ideas evolve and converge into solutions—and design incompatibilities are fixed. As such, iterations can be either beneficial or counterproductive to the PD performance. Iteration is opportune when activities are repeated for idea refinement that contribute to the reduction of technical, schedule, and budgetary risks. In this case, iteration is necessary and should be at least anticipated, if not planned. Meanwhile, counterproductive iteration is akin to the rework of manufacturing processes, as activity repetitions are considered waste and a major source of schedule and budgetary risks, and should be avoided if possible [6, 7].

In an effort to cope with iterations, it is well accepted within the PD body of knowledge that, unnecessary iteration should be minimized (e.g., [8, 9]) and required iterative loops be accelerated (e.g., [3, 10]). Building on this literature, we refer to iteration management as the set of pre-development activities aimed at designing and

evaluating plans (schedules) for managing the many uncertainties involved in the interdependencies of PD tasks. The design of a plan includes the identification of iterative loops, removal of unnecessary iterative loops, and definition of strategies for accelerating the execution of required iteration loops, which has activity overlapping as the well-recognized strategy to this aim. However, this research attempts to complement the overlapping strategy by adopting iteration front-loading as the main strategy to accelerate the overall PD process execution. Meanwhile, plan evaluations include the impact assessment of iterative loops on process performance in terms of lead time statistics. Thereby, we are able to quantify the expected benefit of front-loading iterations not only in terms of the expected value of project lead time but also in terms of the least project performance variance. That is, our goal is to identify the iterative process structure that yields the best performance with the least variation, leading to a more convenient process structure, which is more predictable, and therefore more manageable. In this way, the management of PD projects has the potential to be significantly enhanced through the use of the systematic procedure described herein.

The structure of this paper is as follows. Section 2 provides a review of previous iteration-based management techniques and models. The research method is briefly described in Section 3. The application of the front-loading procedure through an example case study is described in Section 4, while the results are discussed in Section 5. Last, concluding remarks and avenues for future research are presented in Section 6.

2. Literature Review

Considering a project schedule a tool for managing PD processes, an ideal PD project schedule can be defined as one that does not include wasteful iterations but highlights the iterative loops that maximize value (contribute to idea refinement and thus risk reduction) and accelerate flow. Accordingly, counterproductive iterations are viewed as minimization targets for research endeavors [6]. To support this objective, researchers have identified factors that lead to unnecessary iterations. Many researchers suggest that wasteful iterations typically stem from poor process architectures, namely, poor activity sequencing [4,11], avoidable changes in information in coupled activities due to poor activity sequencing or information management problems (e.g., modifications in design objectives, wrong execution times, poor communication and coordination) [6], and poorly timed or too frequent design reviews [12]. Thus, iteration management demands the study of PD activity sequencing by looking at activity interrelationships that define the process structure and the way tasks interact and perform.

Traditional project schedule management techniques, such as Program (or Project) Evaluation Review Techniques (PERT) and the Critical Path Method (CPM), are known to be effective in managing and sequencing networks of activities that consider precedence relationships and activity durations. One of the key limitations, however, is that these techniques provide a schedule that does not account for iterations (the PD process structure is not fully considered) and thus, the schedule reliability is diminished. The inadequacy of these techniques for coordinating PD activities may be attributable to the fact that they were designed to coordinate networks with simpler structures, e.g., networks without probabilistic task iterations.

Meanwhile, there are other efforts to support the management of iterative structures based upon the Design (or Dependency) Structure Matrix (DSM), which is a technique for representing and analyzing complex networks of activities in a square matrix format [6]. The DSM technique partitions the process structure to identify separate "chunks" of activities that are independent and can be processed in parallel, dependent activities that require sequential processing, and "blocks" of coupled activities, that are suggested to be performed iteratively or concurrently. As a result, the DSM not only allows the visualization of the process structure but also improves it, by reducing the number or scope of loops due to ill-defined activity sequencing. It is important to emphasize at this point that the activity sequencing typically occurs at a chunk or block level, i.e. activities within coupled blocks are arbitrarily left in the original order [8] and therefore, questions regarding the management of activities within iterative loops still remain unanswered. Moreover, the DSM does not directly support stochastic modeling [4], which is necessary to evaluate the interaction of loops and how they affect project performance.

In an effort to overcome these and other limitations of the DSM, several researchers have attempted to integrate the original DSM methodology with other techniques to provide insight on how to schedule tasks within iterative loops [8]. Smith and Eppinger [13] developed an analytical DSM extension that allows the computation of overall timing of coupled activities by finding the expected reward (project lead time) of a reward Markov chain. This model, however, involves fixed activity durations and fixed number of iterations. Meanwhile, Chen, Ling, and Chen [14]

considered the learning effect by adding a coefficient, α_i, assessed by the project manager to predict the activity duration for subsequent iteration *i*. In this model, the scheduler still needs to know *a priori* the number of iterations for each activity to compute the overall project duration. Using a different approach, Wang, Liu, and Liao [15] proposed a simulation-based scheduling method in which the input information—the process structure—is provided by the DSM. In their method, Wang et al. [15] assumed that iterations are independent and occur only once. Similarly, Browning and Eppinger [6] developed a method based upon the DSM and Monte Carlo simulation. Although, their method provides an analysis of design iterations in a more generalized project network, one of the key limitations is that the model does not account for all successive feedforward rework [5], leading to less accurate project performance estimations in terms of mean and standard deviation. Cho and Eppinger [5] extended Browning and Eppinger's [6] method by considering the probability distribution of lead-times in stochastic, iterative, and resource-constrained project networks. While this latter model accounts for more characteristics of the process being managed (e.g., feedforward reworks and resource constraints), it does not account for activity costs, and explicit information regarding the criticality of iterative loops is still not provided.

Still, the above-described models fail to consider additional process structure permutations, in particular within blocks, to further investigate if the PD project performance can be improved by either reducing the number of unnecessary loops or changing the scope of necessary iterations within blocks. Instead, it appears that the preferred strategy to speed up the execution of iterations has been overlapping. There are other strategies, albeit less explored in the literature, that can be used to accelerate the process as well. In particular, this work is focused on front-loading that is defined as the shifting of identification and solving of problems to earlier phases of the PD process [10]. Thompke and Fujimoto [10] proposed two front-loading approaches (project-to-project knowledge transfer and rapid problem solving through technology leverage), but they also suggested that supplier relationships, overlapping (earlier starts of problem-solving cycles), and optimal partitioning are other potential approaches to upstream shifts in problems-solving. We draw attention to the latter approach. To this aim, we use an analytical-based framework to design and evaluate project schedules for iterative processes developed by [16]. The core of this framework lies in the streamlined interface between a technique to improve the process structure (DSM) and another used to accurately predict iterative project performance (Graphical Evaluation and Review Technique, GERT). The DSM is used to identify sets of iterative loops and then, we propose a permutation procedure based upon a binary ordering algorithm to front-load iterations within each loop set. Following this, we use GERT from a matrix-based approach to evaluate if the iteration front-loading actually yields to a project performance improvement. The next section highlights major issues regarding the research procedure followed.

3. Research Method

The case study approach was adopted to demonstrate potential benefits of front-loading in PD projects. Scholars typically study PD processes by conducting case studies as empirical data is obtained to build theory about complex relationships about the PD process structure and performance [17]. As a result, within the PD landscape, there are published cases studies that involve in-depth contextual analysis akin to the context of this research. Therefore, there is a high likelihood of obtaining secondary data for our model despite the differences in the solution approach used to address the research problem. Indeed, the data-collection method that seemed most appropriate for this study was to collect project data sets from published case studies that approached the research problem differently, as there was no direct access to information of an ongoing PD project. Using these data sets, our proposed method to front-load iterations was validated and compared with the results of the modeling efforts developed in the published case studies. In this paper, however, our discussions are confined to a single PD project example.

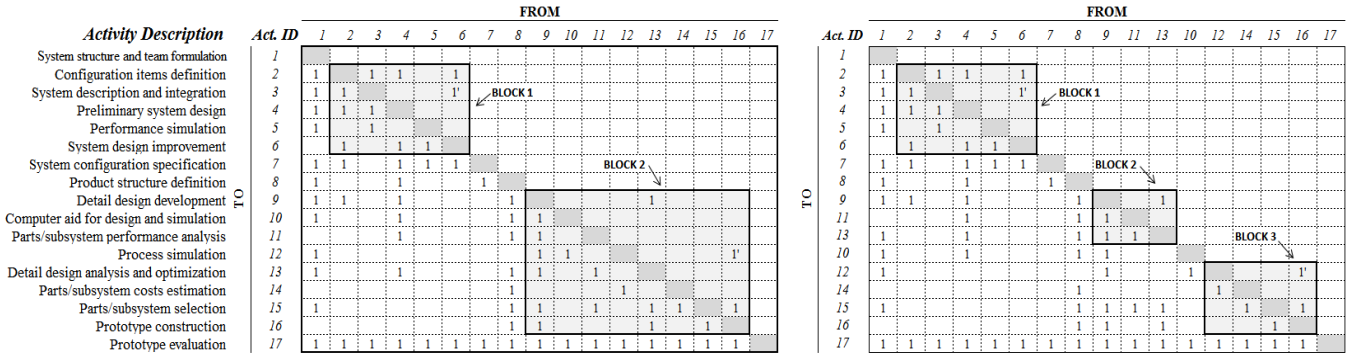
4. Case Study

4.1 Background on the case study

The selected case study comes from Wang and Lin [11]. In their simulation-based model, the performance of a development project from the arms industry is considered. The data set from the battle tank simulator project contains information about the duration and probability of reoccurrence of 17 activities, 73 activity relationships, and at least 7 iterative loops. With this information, the authors assessed three different project structure scenarios to estimate the probability density function of the project completion time and the risk of being late according to a specified target of 78 weeks. However, their modeling approach is slightly different than ours, in that differences of project structures are centered on altering activity overlapping patterns rather than activity sequences. Thereby, the project data from the scenario that considered no overlapping is used to demonstrate the iteration front-loading procedure.

4.2 Application of the iteration front-loading procedure

Figure 1a shows the precedence matrix (PM) of the battle tank simulator project in which activities, precedence relationships (entries below diagonal), and information feedbacks (entries above the diagonal) are displayed. From the PM, two blocks of coupled activities containing seven potential iterative loops in total are identified. To further investigate possible improvements in the process structure, e.g., iterative loop scope reduction, we applied a Boolean-algebra based partitioning procedure developed in [16]. A partitioned PM is the so-called DSM, and when activities within blocks are left in their original order, we refer to this matrix as the conventional DSM (Figure 1b).



(a) Precedence Matrix (PM) (b) Conventional DSM

Figure 1: Matrix representation formats of the battle tank simulator project

Although the structure suggested by the conventional DSM (Figure 1b) still contains seven feedback marks, there are two major differences in the arrangement of these iterative cycles that suggest possible process performance improvements. First, feedback marks are grouped into three blocks instead of two. That is, the iteration cycles emanating from activity 16 (see marks above the diagonal in column 16) are no longer influenced by upstream iterations due to the interactions (information exchanges) between activities 9 and 13. Accordingly, the total lead time with the conventional DSM is expected to be less than the one with the PM project structure. Second, the scope of the iterative cycle due to coupled activities 9 and 13 is reduced because activities 10, 11, and 12 are no longer involved in this loop. An iterative cycle scope or size reduction is visually identified, in the matrix representation, when the feedback mark is leftward shifted, i.e., the feedback mark is closer to the diagonal. Following this logic, we examined each iterative block containing more than three activities, and evaluated if we could first further eliminate feedbacks and then, if we could either shift feedback marks closer to diagonal or to the front-end diagonal.

In explaining the permutation procedure within blocks, a distinction of feedback marks should be made. In Figure 1, notice that there are feedback marks represented by either 1 or 1'. The former represent *irreducible* feedbacks while the latter represent *pseudo* feedbacks. An irreducible feedback is a mark that cannot be removed, unless it is torn, because there is a coupled relationship between activities. For instance, activity 2 provides information to activity 3, but also activity 3 updates information to activity 2. Regardless of the sequence defined between these coupled activities, assumptions about the downstream activity are unavoidable and possible upstream updates remain probable until the project progresses to the next block. At this point, it is important to emphasize that, in a broad sense, a feedback mark exists because an assumption was made by the time an upstream activity had to be performed, e.g., there was an unknown parameter value so that an educated guess was made. Thereby, feedback marks carry information unknowns that, sometimes, can be avoided if activities are sequenced differently (e.g., inverting their order). We refer to avoidable feedback marks as *pseudo* feedbacks which is the case of the matrix elements (row, column) 3, 6 and12, 16 from Figure 1a.

Accordingly, in the permutation within blocks procedure *pseudo* feedback marks are considered first potential candidates for a feed-forward modification. This process begins by identifying blocks formed by three or more activities. For each identified block candidate, two sets of sorting criteria are determined: rows are associated with feedback and with feed forward sorting criteria. The goal is to move downward the most populated rows while shifting to the far left the most populated columns, i.e., iteration front-loading. The evaluation criteria are based upon the Askin and Standrige's [18] binary ordering algorithm, in which row and column sorting occurs in a two-stage operation: first, sorting occurs in a descending order according to feed forward criteria, and then in ascending

order according to feedback sorting criteria. Following this, an evaluation of the remaining pseudo feedback marks is performed. That is, those activities involved in the *ij* feedback marks and which element *ji* is zero (or blank), are considered for a further re-arrangement by shifting row/column *j* before row/column *i*. This re-arrangement alternative is held only if the total number of feedback marks decreases; otherwise, the permutation is not considered and the next *pseudo* feedback mark is evaluated for a possible re-arrangement. This operation is continued until all potential *pseudo* feedback marks are exhausted. In this work, the resultant DSM matrix with activities re-ordered within blocks is referred to as the *extended* DSM, shown in Figure 2. The differences among the process structures can also be easily discerned in a GERT-type activity network, as depicted in Figure 3, in which trivial feed-forward marks were removed (only immediate precedence relationship were left), irreducible feedback marks are shown in red, and pseudo feedback marks are in blue.

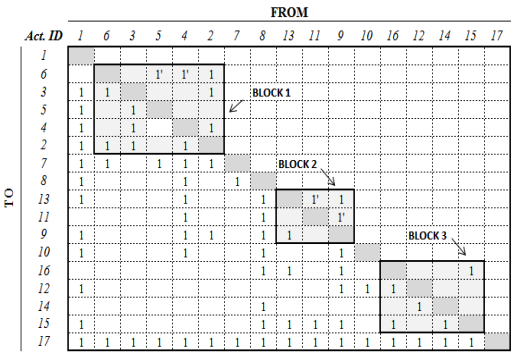


Figure 2: The extended DSM

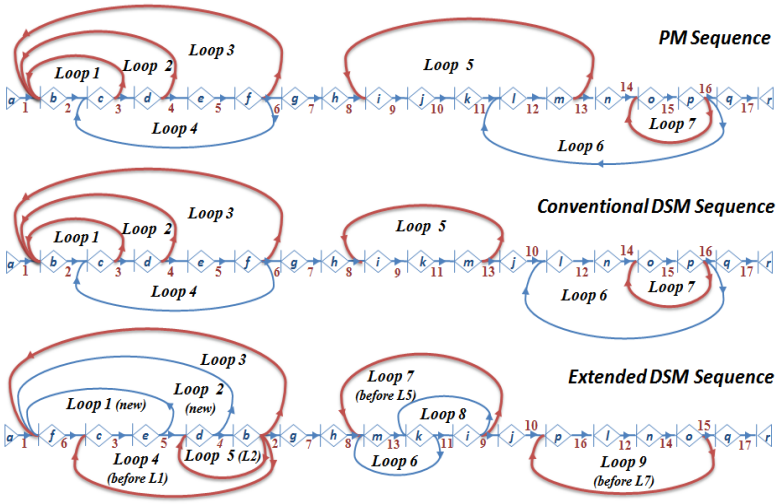


Figure 3: GERT-type network comparison

The first noticeable observation from the extended DSM process structure is that the total number of iterative loops increased to nine. However, fewer loops do not necessarily yield a better process performance [8], as the loop size and location also bear on project lead time. It is also interesting to note that in block 3, a pseudo-feedback mark was indeed eliminated, but at the expense of augmenting the scope of the irreducible loop 9 (previously loop 7 in the conventional DSM sequence from Figure 3). While reducing the scope or size of loops is often associated with faster or less costly activity repetitions, iterations at the back-end are also linked with slower and more costly executions. This suggests that having less iterations at the back-end of the process may, indeed, improve the process performance. This is particularly true in this example and will be later demonstrated with data. Moreover and referring to block 2, notice that while the size of loop 7 (previously loop 5 in the conventional DSM structure) cannot be reduced given that it is a maximal loop, i.e. a set of nodes connected by a closed path that cannot be extended from either end, the consequence of inverting the order of the emanating and incident activities (9 and 13) implied the creation of two *pseudo* feedbacks (loops 6 and 8). Similarly in block 1, the population of loops increased. Another observation worth pointing out is that while the total number of feedback marks increased, these increments occurred only in the first two iterative blocks, i.e., iterations in the extended DSM are front-loaded.

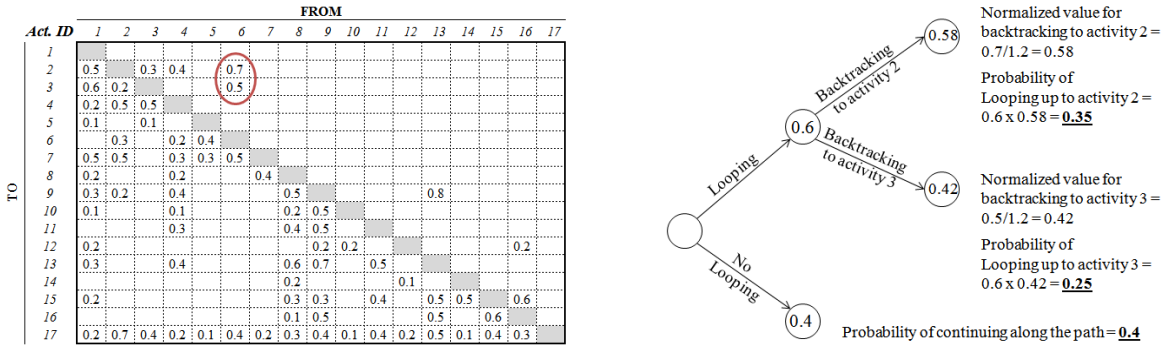
Based on the previous observations, the following questions arise. Will the elimination of pseudo-feedback marks have a positive effect on project performance in the third iterative block? Will front-loading iterations, even though they produce more pseudo-feedback marks, accelerate the execution of the iterative block? Overall, will front-loading iterations accelerate the project execution? To answer these questions, the interaction of loops along the project path must be considered to determine if the expected project performance with a 9-loop structure is better or worse than the performance of the project with a 7-loop structure. Thus, the expected project performances are estimated and compared for the three different process structures. The behavior parameters in terms of activity durations and probabilities of occurrence are described next.

4.3 Treatment of data for project performance estimations

The data set for this example case study comprised activity durations and rework effort, both described in terms of optimistic, most likely, and pessimistic time values. With this information, the triangular distribution was used for

estimating activity durations with rework times considered. Aligned with previous iteration-aware models [4, 5], rework effort was decomposed into activity learning curve and rework impact percentages (task % that must be reworked or revised). In addition to the rework effort, rework probabilities were necessary for computing activity rework total durations. For this example case, a rework probability matrix was available, in which numerical values were aligned with this research: values below the diagonal represent the probability of reworking upstream activity *i* given that a change in the input from downstream activity *j* occurred and values above the diagonal signify the probability of iteration starting from activity *j*. However some of these values had to be normalized due to the assumptions of the analytical method selected for estimating the project lead time. This is further discussed.

GERT was used to determine the overall network transmittance to compute the project lead time, in which the interaction of intermingled iterative loops are considered without *a priori* assuming a fixed number of iterations per loop. The estimations of the number of iterations are based on the probabilities of occurrence (rework) (shown in Figure 4a). However, one of the underlying assumptions of GERT is that only one emanating arrow can be taken from a given node. For example, consider node *f* from the conventional DSM structure depicted in Figure 3. GERT assumes that the path can be either traversing the arrow that leads to node *g* or backtracking the project to either node *b* or *c* but not both. Thus, probabilities of occurrence from these three possible paths must sum one. Unfortunately, the probability of rework in the original data set did not comply this assumption (see numbers in oval from Figure 4a), and therefore, these quantities had to be normalized as shown in Figure 4b.



(a) Rework probability (RP) matrix (adapted from [11]) (b) RP normalization example

Figure 4. Rework probabilities

5. Results and Discussion

Table 1 highlights the main changes in the value of parameters for the each process structure that yield different performance outcomes in terms of lead time expected values and standard deviations. Among the salient parameter changes resulting from the extended DSM are the following. The probability of iteration decreased, in particular for loop 5 (Loop 7 in the extended DSM sequence). Accordingly, the expected number of iterations per loop also diminished, which also affected the total lead time per loop. Table 1 also presents the performance breakdown per iterative block, allowing the impact quantification due to activity re-arrangements within blocks. From this, several interesting observations can be made. First, it is noticed that, while the critical path remained invariant with the original and conventional DSM structures, a significant reduction is observed with the extended DSM structure. The 15-unit difference is due to the fact that activities 10, 12, and 16 are no longer critical; therefore, these activities were naturally overlapped. Thus, the parallel execution of these activities, which are involved in iterative block 3, may explain the negligible looping effect in terms of time. Moreover, the difference in the expected lead time of block 2, and referring only to the DSM structures, is the result of inverting the starting and finishing activities. Even though this created more loops, the total duration in block 2 was significantly reduced. This is because the value of the activity coupling strength is lowered in the opposite direction. However, there were other instances in which loop probabilities decreased due to the normalization required to comply GERT assumptions, as mentioned in section 4.2. Furthermore and while the least improvement is obtained in iterative block 1, it surfaced another potential benefit of the activity re-arrangement within blocks. By inverting the order of coupled activities, we are deciding what assumption to make –as feedback marks are considered assumption carriers [9]. Activity 2 was the starting activity of all the irreducible loops from block 1 of the extended DSM. Not surprisingly, activity 2 was forced to make three different assumptions regarding the output of activities 3, 4, and 5. But when the execution activity 2 is delayed, as suggested in the extended DSM, the result is that three different activities—3, 4, and 5—still have to make an assumption, but all related to the output of activity 2. Interestingly, it appears that delaying the

assumptions to be made by activity 2 and including two more design revisions (loops 1 and 2 in Figure 3), actually helped to accelerate the execution of the project. Overall, the impact of the iterative blocks, as defined in the extended DSM structure, reduced the overall project lead time, also leading to less variation.

Table 1. Comparison of parameter values for each process structure

Network Attributes & Performance Values	***Original***	***Conventional DSM***	***Extended DSM***
Representative Structure Characteristics			
Number of chunks of nested loops	2	3	3
Number of loops	7	7	9
Representative Parameter Changes			
Iteration probability interval values	0.2-0.8	0.2-0.8	0.1-0.6
Iteration probability L5	0.8	0.8	0.3
Number of iteration interval values	0.625-8	0.625-4	0.2-2
Loop length interval values	1.05-15.24	1.05-14.56	0-5.66
Expected Project Performance Results			
Critical Path Length (CPL)	**60.2**	**60.2**	**45.5**
Iterative Block breakdown			
Block 1 (B1)	8.6	8.6	7.0
Block 2 (B2)	128.9	58.2	7.3
Block 3 (B3)	NA	7.0	0.1
Total Loop Lead Time (LLT) = B1 + B2 + B3	**137.6**	**73.8**	**14.4**
Total Project Lead Time = CPL + LLT	**197.8**	**134**	**59.9**
Std. Dev.	**134.3**	**66.28**	**11.9**

It is also appealing to compare the iteration front-loading strategy with other alternatives to accelerate the process, e.g., activity overlapping and loop deletion. Table 2 summarizes the main outcomes of the modeling techniques that deployed different work policies and are now compared with the 78-week project completion target, Wang and Lin's [11] modeling results. From these results, the strategies of iteration front-loading (extended DSM), aggressive overlapping (Wand and Lin's [11] model), and loop deletion (conventional DSM process structure without loop 5) would make the 78-week target for project completion accomplishable—expected lead-times are 59.88, 75.53 and 75.77 weeks, respectively. However, according to the data, the process structures with either iterations front-loaded or without loop 5 suggest more predictable projects, as both have the lowest standard deviation values (approximately 12). Therefore, these project structures may be more manageable. The difference is that deleting loop 5 may be unrealistic or unfeasible, but controlling more iterations may be more demanding (with the iteration front-loading strategy). Yet, for this case, the iteration-front loading appears to be superior as this process structure yields the lowest completion time and standard deviation without tearing irreducible loops.

Table 2.Modeling techniques and acceleration strategy comparison

			Project completion time		
Approach	***Management Techniques***	***Strategy to reach target (improve the PD performance)***	***Mean***	***Deviation from target***	***Std. Dev.***
	Target		***78***	***0***	***NA***
Analytical -based	DSM-GERT	Iteration front-loading (Extended DSM sequence w/all loops)	59.9	-23.23%	11.91
	DSM-GERT	Loop deletion (Conventional DSM sequence without loop 5)	75.78	-2.85 %	12.44
Simulation-based	Wang & Lin [11]	Conservative overlapping (with PM activity sequence)	83.7	7.31 %	38.15
	Wang & Lin [11]	Aggressive overlapping (with PM activity sequence)	75.53	-3.17 %	31.78

In terms of study limitations, given that it was based on secondary data, the process structures thus far suggested still require management evaluation to determine their feasibility. Also, performance was only evaluated in time units. It would be interesting to determine the cost of front-loading the process and identify the project conditions (e.g., loop structure and parameter values) in which iteration front-loading is more versus less convenient. Moreover, the activity permutation within blocks procedure does not guarantee actual improvements in the process performance given that ordering activities with the least number of loops does not necessarily minimize project execution times [8]. Finally, this study was based on a single project. Overall, a critical next step is further testing of the iteration front-loading framework with multiple and ongoing projects to further validate its effectiveness.

6. Conclusions

The application of this case study demonstrated that, the process structure with more loops (extended DSM structure) yielded a better project performance than the structure with fewer loops (original and conventional DSM process structures). This appears due to the fact that, in the extended DSM, loops were located frontward along the project path, suggesting that, in this particular case, iteration front-loading is a superior strategy than loop deletion or overlapping critical-path activities. It was also demonstrated that the least number of feedback marks in a process structure does not necessarily promote faster project performances. These empirical findings suggest that further research is needed on the impact assessment of iterative loops. Iteration management should also include procedures to identify which assumptions (feedbacks) help to reduce technical, schedule, and budgetary project risks. The application of the iteration front-loading procedure to the case study demonstrates that iteration management enhances the estimation breakdowns per iterative block, while at the same time enabling the identification of the process structure with the least variation. Our current findings suggest that iteration management will be enhanced to the extent that more partitioning procedures for iteration front-loading are developed. After all, the rationale of the iteration paradigm is to plan product design iterations to perform the product and manufacturing design processes more effectively and efficiently by accelerating the required design iterations while minimizing counterproductive iterations in terms of engineering changes.

References

1. Verworn, B., Herstatt, C., and Nagahira, A., 2008, "The fuzzy front end of Japanese new product development projects: impact on success and differences between incremental and radical projects," R&D Management, 38(1), 1-19.
2. Sobek, II, D. K., 1997, "Principles that Shape Product Development Systems: A Toyota-Chrysler Comparison," Ph.D. dissertation, The University of Michigan.
3. Smith R. P., and Eppinger, S. D., 1997, "Identifying controlling features of engineering design iteration," Management Science, 43(3), 276-293.
4. Browning, T. R., and Eppinger, S.D., 2002, "Modeling impacts of process architecture on cost and schedule risk in product development," IEEE Transactions on Engineering Management, 49(4), 428-442.
5. Cho, S. -H., and Eppinger S. D., 2005, "A Simulation-Based Process Model for Managing Complex Design Projects," IEEE Transactions on Engineering Management, 52(3), 316-328.
6. Browning, T. R., 1998, "Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development," Ph.D. dissertation, Massachusetts Institude of Technology.
7. Denker, S., Steward, D. V., and Browning, T. R., 2001, "Planning Concurrency and Managing Iteration in Projects," Project Management Journal, 32(3), 31-38.
8. Steward, D. V., 1981, System Analysis and Management: Structure, Strategy and Design, Petrocelli Books Inc., New York.
9. Black, T. A., Fine, C. H., and Sachs, E. M., 1990, "A Method for Systems Design Using Precedence Relationships: An Application to Automotive Brake Systems," Working Paper, M.I.T. Sloan School of Management.
10. Thomke, S., and Fujimoto, T., 2000, "The Effect of Front-Loading Problem-Solving on Product Development Performance," Journal of Product Innovation Management, 17, 128-142.
11. Wang, J., and Lin, Y. -I., 2009, "An overlapping process model to assess schedule risk for new product development," Computers & Industrial Engineering, 57(2), 460-474.
12. Ha, A. Y., and Porteus, E. L., 1995, "Optimal Timing in Reviews in Concurrent Design for Manufacturability," Management Science, 41(9), 1431-1447.
13. Smith, R. P., and Eppinger, S. D., 1997, "A Predictive Model of Sequential Iteration in Engineering Design," Management Science, 43(8), 1104-1120.
14. Chen, C.-H., Ling, S. F., and Chen, W., 2003, "Project scheduling for collaborative product development using DSM," International Journal of Project Management, 21, 291-299.
15. Wang, W.-C., Liu, J. -J., and Liao, T. -S., 2006, "Modeling design iterations through simulation", Automation in Construction, 15, 589-603.
16. Martinez, C. 2010, "An Analytical Methodology for Designing and Evaluating Project Schedules for Iterative and Uncertain Processes," Ph.D. dissertation, Texas Tech University.
17. Creswell, J. W., 2003, Research Design Qualitative, Quantitative, and Mixed methods Approaches, 3rd Eddition, SAGE Publications, Thousand Oaks.
18. Askin, R. G., and Standridge, C. R., 1993, Modeling and Analysis of Manufacturing Systems, Wiley, N.Y.