A System Architecture-based Model for Planning Iterative Development Processes:
General Model Formulation and Analysis of Special Cases

Jay Jootar, Steven D. Eppinger

Abstract – **The development process for complex system is typically iterative in nature. Among the critical decisions in managing such process involves deciding how to partition the system development into iterations. This paper proposes a mathematical model that captures the dynamics of such iterative process. The analysis of two special cases of the model provides an insight into how such decision should be made.**

Keywords – **product development, iterative process, risk management, decomposition**

I. Introduction

An important characteristic of the development process for complex systems is the iterative nature of the activities (Whitney 1990). This is primarily the result of complex, often times cyclical, relationships of elements of the system. In these circumstances, sometimes there remains significant uncertainty even after the early system design phase. Interim validation and verification are necessary to test the assumptions of the system as we build it, and hence the iterative nature of the development process.

This paper seeks to capture the dynamics of iterative development processes in a formal model. In particular, we present a mathematical model that describes the way the process execution affects the process performance. Although many such models exist in the literature (Ha & Porteus 1995, Smith & Eppinger 1997a, Smith & Eppinger 1997b, Krishnan et al. 1997, Loch & Terwiesch 1998), the model proposed in this paper extends prior models by including architectural dependencies in the system and yet is tractable enough that insightful analysis can be undertaken.

The paper begins by describing the general model of system and iterative development. Due to space limitation, the analysis of two simple, yet illustrative, special cases of the model are provided in the next section. The emphasis of the analysis is on demonstrating the fundamental trade-off of the model and how model parameters affect the resulting execution plan. The paper ends with the summary of the results and future research directions.

II. Model Formulation

A. General Description

We model a critical stage in the development process when architectural activities have been conducted to the extent that the system is decomposed into modules, with a mapping of functions to these modules and yet there remains uncertainty in some aspects of the system that has to be dealt with by building certain parts of the system. This calls for an iterative mode of development in which knowledge is gained from building parts of the system and is then used to reduce uncertainty in the system development.

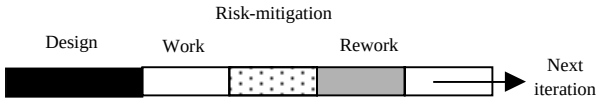

Fig. 1. Iterative Process

The nature of the process requires a partitioning of the development of the system modules into iterations. In successive iterations, we grow the system by developing more and more modules, while making necessary changes along the way as we learn more about the system. In the **nominal work phase**, we first develop the modules allocated to that iteration and integrate these modules with the rest of the system that has been developed so far. Then we test the resulting system against its intended functions during the **risk-mitigation phase**. The information revealed in this phase might necessitate rework of these modules in the **rework phase** before we proceed to the next iteration. The total amount of work depends on how we partition the modules into iterations, which is the decision resulting from the model and the major topic of the next sections.

B. System Partitioning

More formally, a system consists of M modules, numbered from 1 to M. The set of modules is called Ω.

Denote the partitioning by $\left(S^1, S^2, ..., S^J\right)$, where S^j represents the set of modules allocated to iteration j.

Define $\hat{S}_j = \bigcup_{i=1}^{j} S_i$ as the set of modules developed up to iteration j.

Define a function as a set of modules. Each function corresponds to the functional requirement that we test during the risk-mitigation phase. Assume there are N functions, with $F_n \subseteq \Omega$, as the set of modules for function n. Given a set of modules we can find the corresponding set of functions whose constituent modules belong to that set. Define the mapping as

$$\Im\left(\hat{S}\right) = \left\{n \mid F_n \subseteq \hat{S}\right\}$$

The set of modules in successive iterations corresponds to the set of functions in successive iterations in the following manner.

$$\Im^j = \Im\left(\hat{S}^j\right) \setminus \Im\left(\hat{S}^{j-1}\right)$$

The variables of concern in each iteration include (1) **risk-mitigation work**, which is assumed to be a fixed constant, denoted L, for each iteration and (2) **rework** which depends primarily on the partitioning of modules as described in the following section.

C. Rework

Rework depends on the partitioning of modules in two major ways. First, rework in an iteration, say j, is defined only for modules that have been developed up to iteration j, i.e. modules that belong to $\hat{S}_j$. In other words, we only need to do rework on modules that have already been worked on. Second, the amount of rework of such modules depends in a non-trivial manner to how we allocate modules to iterations described as follows.

In a decomposition of a complex system, we also have to define interfaces that link the modules together to become a system. These interfaces represent the assumptions from the point of view of developing a certain module. They are the constraints that are given for the problem to be solved. When we deal with a complex system these interfaces are uncertain and subject to change. They are the driver of rework in the development. In this model, we define an interface as a design assumption subject to change.

The amount of rework of a particular module in a given iteration depends on interfaces that actually change in that iteration. For simplicity, we assume that it is the sum of rework resulting from change in each interface. Define interface change for each iteration as a zero-one random variable. Assume there are Q interfaces for the system in question.

$$\tilde{R}_m^j = \sum_{q=1}^{Q} d_{mq} \cdot \tilde{I}_q^j$$

We call d_{mq} the **impact parameter** of the module. Note that $\tilde{I}_q^j$ is fully described by $P_q^j = \Pr\left(\tilde{I}_q^j = 1\right)$

Next, we need to define how interface change happens. First, we need to observe that interfaces have a strong relationship with the functions of the system. Since they specify how the modules work together, they have a direct effect on some functions of the system. Due to the complexity of the system, this is not known in advance, we know whether a particular assumption, or interface, works or does not work only when we test the system against its functional requirements.

From this observation, the probability of change of a particular interface in an iteration depend on the functions corresponding to that iteration. For simplicity, we assume that the probability P_q^j is the sum of **probability parameters** p_{qn} of all functions in iteration j.

$$P_q^j = \sum_{n \in \Im^j} p_{qn}$$

D. Work-minimization Model

In this paper, we focus our attention on the optimization of the total expected work, which include the risk-mitigation work and rework. The dynamic programming formulation of a work-minimization model can be described as follows

$$W(\hat{S}) = \min_{S \subseteq \hat{S}^C} \left[L + \sum_{m \in S \cup S} E\left[\tilde{R}_m\right] + W(\hat{S} \cup S) \right]$$

$$= \min_{S \subseteq \hat{S}^C} \left[L + \sum_{m \in S \cup S} E\left[\sum_{q=1}^{Q} d_{mq} \cdot \tilde{I}_q(\hat{S}, S) \right] + W(\hat{S} \cup S) \right]$$

$$= \min_{S \subseteq \hat{S}^C} \left[L + \sum_{m \in S \cup S} \sum_{q=1}^{Q} d_{mq} \cdot P_q(\hat{S}, S) + W(\hat{S} \cup S) \right]$$

$$= \min_{S \subseteq \hat{S}^C} \left[L + \sum_{m \in S \cup S} \sum_{q=1}^{Q} d_{mq} \cdot \sum_{n \in \Im(\hat{S} \cup S) \setminus \Im(\hat{S})} p_{qn} + W(\hat{S} \cup S) \right]$$

Given $W(\Omega) = 0$, Find $W(\varnothing)$

III. Model Analysis

The purpose of this section is to develop the understanding of the basic trade-offs in the model to the reader. We will focus our attention on two special cases of the model which are relatively easy to understand yet illustrative of such trade-offs.

A. Modular System with Uniform Impact

The system to be considered in this section has the following properties

1. There is one interface subject to change.
2. The mapping between function and module is one-to-one, i.e. a module corresponds to a function.
3. All the modules have the same impact parameter, denoted *d*.

We assume that the module and function has been indexed in such a way that module *m* belongs to function *m* for *m*=1, …, *M*. In this model, the parameter of interest therefore includes *L*, *d*, p_1, …, p_M. From this, we can rewrite the defining equation for the dynamic programming formulation as follows.

$$W(\hat{S}) = \min_{S \subseteq \hat{S}^C} \left[L + \left(\sum_{m \in S \cup S} d \right) \cdot \left(\sum_{m \in S} p_m \right) + W(\hat{S} \cup S) \right]$$

Note that subscript *q* is omitted because there is only one interface subject to change. For a given partitioning, define the following quantities which correspond to each iteration.

(1) $P^j = \sum_{m \in S^j} p_m$ the probability that rework will be needed in iteration j.

(2) $D^j = \sum_{m \in S^j} d = \left| S^j \right| \cdot d$ the incremental amount of potential rework added from modules in iteration j.

The total amount of potential rework is the sum of the product of these quantities from iteration 1 to iteration j. The total work can also be expressed as

$$\sum_{j=1}^{J} L + P^j \cdot \left(\sum_{i=1}^{j} D^i \right)$$

The first issue we will explore is that of the structure of optimal policy. The following theorem shows the structure of the optimal solution when viewed from the perspective of probability parameters.

Theorem 1: In the optimal policy for an additive probability function, let $p^j_{\min}$, $p^j_{\max}$ be the minimum and maximum probability parameters, respectively, among those of functions allocated to iteration *j*. Then,

$$p^j_{\min} \geq p^{j+1}_{\max}$$

Proof: Proof by contradiction. First assume that $p^j_{\min} < p^{j+1}_{\max}$ in an optimal solution. Then, we swap the two modules corresponding to these two probability parameters. Subtract the new objective from the original one to obtain $\left(p^{j+1}_{\max} - p^j_{\min} \right) D^j$, a positive number, contradicting the hypothesis that the original partitioning is optimal. Hence, $p^j_{\min} \geq p^{j+1}_{\max}$ in an optimal solution.

From this theorem, we can reduce the complexity of the problem substantially by first sorting the modules from the highest probability to the lowest probability, then determining the splitting point that gives the minimum total work. The problem can be reformulated for the pre-sorted modules as follows.

$$W(\hat{m}) = \min_{m > \hat{m}} \left[L + (m - \hat{m}) \cdot d \cdot \left(\sum_{i=\hat{m}+1}^{m} p_i \right) + W(m) \right]$$

This problem in turn could be translated into a shortest-path problem for a network defined by nodes that correspond to each module/function arranged in sequence and arcs that correspond to the amount of work resulting from allocating the module between the nodes in a particular iteration.

In addition, this theorem also suggests the way the partitioning should be done to minimize the amount of

work. From this theorem, the best partitioning is obtained by taking care of the modules/functions with high probability parameters first. This corresponds to the rule of thumb in system engineering that we should take care of the tough issues, the issues that are most likely to cause changes, first.

As an example, we have generated an example with 100 modules/functions where L = 1, d = 1. The probability parameters of the functions are generated randomly so that the sum is unity. As explained above, modules are first sorted from highest probability parameters to the lowest probability parameters. Then, we use the shortest-path algorithm to find the best policy. The following chart displays the optimal solution in terms of iteration quantities.

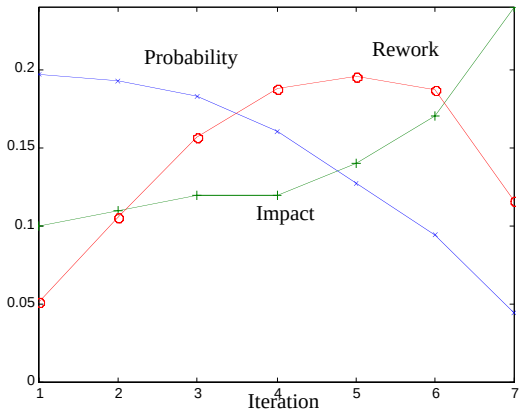

Fig. 2. Iteration Parameters in Optimal Solution

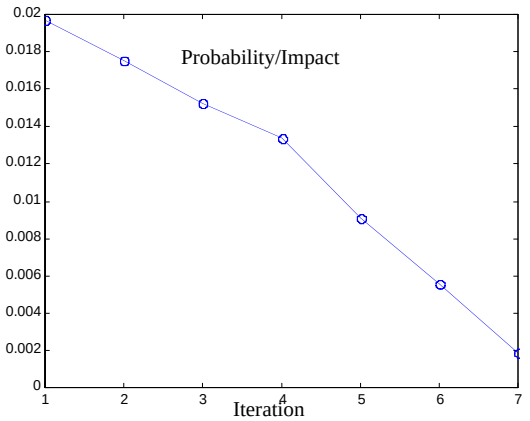

Fig. 3. Iteration Ratio in Optimal Solution

As can be observed from the chart, there is a clear trend of the iteration probability and the iteration impact. In the optimal policy, iteration probability has a downward trend, whereas iteration impact has an upward trend. In addition, the ratio of iteration probability to iteration impact seems to be decreasing. The following theorems provide the proof for these trends.

Theorem 2: In the optimal partitioning, $\frac{P^j}{D^j} \geq \frac{P^{j+1}}{D^{j+1}}$

Proof: Use the swapping argument. First, assume that in an optimal policy, $P^j/D^j < P^{j+1}/D^{j+1}$. Next, swap the modules in these two iterations. Subtract the objective of the resulting partitioning to that of the original one to obtain $P^{j+1} \cdot D^j - P^j \cdot D^{j+1}$, a positive quantity. This contradicts our assumption that the original partitioning is optimal. Therefore, in an optimal solution, $P^j/D^j \geq P^{j+1}/D^{j+1}$.

Theorem 3: In the optimal policy, the following conditions are true.

(1) $D^j - D^{j+1} \leq d$

(2) $P^j - P^{j+1} \geq -p_{\max}^{j+1}$

Proof: (1) We first prove that in an optimal partitioning, $d_m/p_m \geq \left(d + D^{j+1}\right) P^j$ for any $m \in S^j$. This can be proved by contradiction. First, assume that the opposite of the above condition is true in an optimal partitioning. Then, we consider the difference in objective between the supposedly optimal partitioning and the one in which m is moved from iteration j to iteration j+1, which, due to the opposite conditioned, is positive, thus contradicting the optimality assumption. Therefore, the condition has to be true for an optimal partitioning. Then, $D^j/P^j \leq \left[d/p\right]_{\max}^j \leq \left(d + D^{j+1}\right) P^j$. The first inequality is due to the fact that the iteration impact and iteration probability are simply the sum of the individual parameters of all modules/functions in the iteration. The second inequality comes from the condition we just proved. Hence, we obtain the result $D^j - D^{j+1} \leq d$. (2) The second condition can be proved in the same fashion. We first prove that $\left(P^j + p_m\right) D^{j+1} \geq p_m/d_m$ for any $m \in S^{j+1}$ using a similar line of argument. Then, $\left(P^j + p_{\max}\right) D^{j+1} \geq \left[p/d\right]_{\max}^{j+1} \geq P^{j+1}/D^{j+1}$. Hence, the property $P^j - P^{j+1} \geq -p_{\max}^{j+1}$.

These theorems show the characteristics of the optimal policy of this type of system. We found that in general, not only do we tackle modules with highest probability parameters first, but we also group them in such a way

that the iteration probability is not increasing by more than the probability parameter and that the iteration impact is not decreasing by more than the impact parameter. This means that, in general, the partitioning results in more of the uncertainty (in terms of iteration probability) handled in the early iterations and results in more of the impact handled in later iterations when the remaining uncertainty is low (after we handle most of that in the early iterations)

B. Nested System

The next system we consider is that of a nested functional mapping, which has the following properties.

1. There is only one interface subject to change.
2. The set of functions represent an increasing set of modules, i.e. the module set of a function is a subset of the module set of another function. Represent this function as 1, 2, …, *M*, as an increasing set of function. $F_m \subseteq F_{m+1} = F_m \cup \{m+1\}$

In this case, functions build upon each other. Compared to the previous case when each function is independent of the others. In this case, a particular function relies on modules of other functions. The first issue, again, is the structure of the optimal policy. The functional mapping seems to suggest that we should build the internal functions and grow the system in the way dictated by the functional mapping. Theorem 4 proves this result in a rigorous manner.

Theorem 4: In the optimal partitioning of modules,

$$m \in \hat{S}^j \rightarrow \{1, 2, ..., m\} \subseteq \hat{S}^j$$

Proof: Use contradiction. In an optimal partitioning, suppose $m \in \hat{S}^j$. Let m_0 <m be the module with the smallest index which is absent from $\hat{S}^j$. Then, we have $\Im(\hat{S}^j) \cap \{m_0, m_0+1, ..., m\} = \varnothing$ due to the nested structure of the functional mapping. Assume that all modules have non-zero impact parameters, we can improve the objective by moving all modules whose index falls anywhere from m0+1 to m to iteration j+1.

Such movement does not change the iteration probability of any iteration before j+1 because such probability can only be the sum of probability parameters of functions whose index is less than m0. This is due to our assumption that m0 is not in $\hat{S}^j$ and to the nested structure of the functional mapping. Next, the movement reduces the iteration impact for those iterations to which the modules that are moved belong. On the other hand, the movement does not change anything from iteration j+1 onward. This therefore results in a decrease of the objective, contradicting the optimality assumption. Hence, we obtain the above result.

From theorem 4, the defining equation for the dynamic programming formulation can be expressed as

$$W(\hat{m}) = \min_{m>\hat{m}} \left[L + \left(\sum_{i=1}^{m} d_i \right) \cdot \left(\sum_{i=\hat{m}+1}^{m} p_i \right) + W(\hat{m}+m) \right]$$

This, similar to the previous case, can be transformed into a network and a shortest-path algorithm can be used to solve the problem.

Similar to the previous case, the sequence of the modules is determined by the problem parameter and the remaining question is how to split them into iterations. Whereas the probability parameters determine the sequence in the modular case, the functional mapping determines the sequence in this nested structure. Probability parameters and impact parameters play no role in determining the sequence in this case.

In a sense, these two cases are similar in that the guiding principle is to partition the modules so that we take most uncertainty out of the design process as soon as possible. The specifics are different due to the structure of the problem. In the previous case, functions are independent in a sense that they don't share modules. In this case, probability parameters determine in which iteration a particular module will reside.

On the other hand, in the nested case, the partitioning of the modules needs to take into account the functional mapping. In particular, the nested structure dictates that the lower level-function, i.e. the functions whose modules are the constituent modules of other functions needs to be taken care of before other higher-level functions.

Because the sequence is dictated solely by the functional mapping, the characteristics of the optimal policy for the nested structure might be different from that of the modular structure, whose characteristics derive from the fact that modules/functions are ordered so that probability parameters are in descending order. To illustrate this, we have generated an example such that the probability parameters of function in the sequence are in ascending order. We clearly see that the trend is the reverse of the previous case.

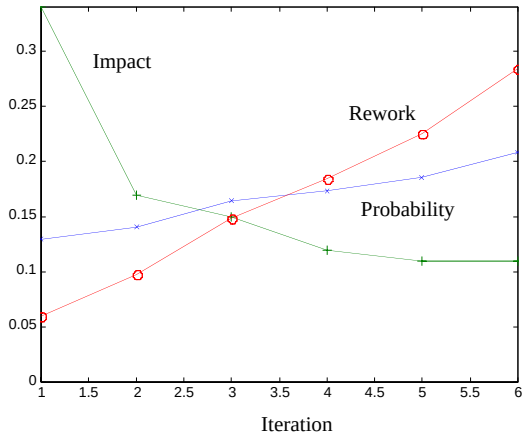

Fig. 4. Iteration Parameters in Optimal Solution

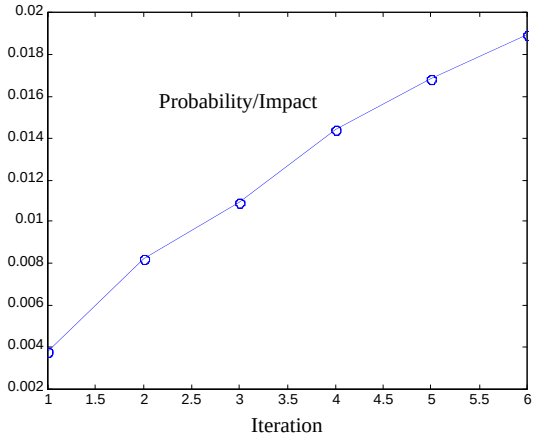

Fig. 5. Iteration Ratio in Optimal Solution

This is the case when the module sequence is solely dictated by the functional mapping. In these circumstances, the iteration quantities do not behave in the same fashion as the previous case. The example generated above has the reverse trend, i.e. iteration probability is lower in the earlier iteration, iteration impact is higher in the earlier iteration, and iteration probability/impact ratio is increasing.

IV. Conclusions

In this paper, we have formulated the model of an iterative process, which is modeled as a process in which a system is built in an incremental manner. The model describes how various types of additional work (risk-mitigation work and rework) depend on the way the system is partitioned into iterations.

We analyzed two special cases of a model that seeks to minimize the total amount of work. We have found that the guiding principle is to partition the system in such a way that we tackle as early as possible the most uncertainty with the least amount of impact.

In the first special case, this results in a policy in which we sequence the modules/functions by their probability parameters. In the second case, this results in a policy in which we sequence the functions from low-level to high-level. These two cases illustrate the effects of functional mapping, probability parameters, and impact parameters in determining the optimal partitioning of modules/functions to minimize the amount of work to be undertaken.

Future research should explore the joint effects of the above model parameters. We are working on a general case of the model and to explore how various parameters jointly determine the optimal policy.

References

[1] Ha, A. Y., E. L. Porteus, “Optimal Timing of Reviews in Concurrent Design for Manufacturability,” *Management Science*, 41, 9 (1995), 1431-1447.

[2] Krishnan, V., S. D. Eppinger, D. E. Whitney, “A Model-Based Framework to Overlap Product Development Activities,” *Management Science*, 43, 4 (1997), 437-451.

[3] Loch, C. H., C. Terwiesch, “Communication and Uncertainty in Concurrent Engineering,” *Management Science*, 44, 8 (1998), 1032-1048.

[4] Smith, R. P., S. D. Eppinger, “Identifying Controlling Features of Engineering Design Iteration,” *Management Science*, 43, 3 (1997a), 276-293.

[5] Smith, R. P., S. D. Eppinger, “A Predictive Model of Sequential Iteration in Engineering Design,” *Management Science*, 43, 8 (1997b), 1104-1120.

[6] Unger, D., “Planning Design Iterations”, SMA Paper, January 2002.

[7] Whitney, D. E., “Des igning the Design Process”, *Research in Engineering Design* (1990)