Optimal information exchange policies in integrated product development

Ali Yassine\textsuperscript{a}, Bacel Maddah\textsuperscript{a} & Nabil Nehme\textsuperscript{b}

\textsuperscript{a} Engineering Management Program, American University of Beirut, Beirut, Lebanon
\textsuperscript{b} Department of Civil and Environmental Engineering, American University of Beirut, Beirut, Lebanon

Accepted author version posted online: 15 Jan 2013. Published online: 29 Jul 2013.

To cite this article: Ali Yassine, Bacel Maddah & Nabil Nehme (2013) Optimal information exchange policies in integrated product development, IIE Transactions, 45:12, 1249-1262, DOI: 10.1080/0740817X.2012.762487

To link to this article: http://dx.doi.org/10.1080/0740817X.2012.762487

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at http://www.tandfonline.com/page/terms-and-conditions
Optimal information exchange policies in integrated product development

ALI YASSINE¹,∗, BACEL MADDAH¹ and NABIL NEHME²

¹Engineering Management Program, American University of Beirut, Beirut, Lebanon
E-mail: ali.yassine@aub.edu.lb
²Department of Civil and Environmental Engineering, American University of Beirut, Beirut, Lebanon

Received February 2012 and accepted November 2012

This article considers information exchange in an Integrated Product Development (IPD) environment. First, a dynamic programming model is formulated that is able to capture upstream partial information flow in a two-activity IPD process. A simple threshold policy is derived that aids the downstream activity in deciding whether to consider or ignore this upstream information as a function of information quality and its associated setup and rework penalties. Then, this formulation is expanded to model analytically, for the first time, information flow in a three-activity IPD process. In this case, the focus is on aiding the midstream activity in deciding whether to consider or ignore partial upstream information, taking into consideration downstream concerns. Because it is difficult to derive threshold policies in this case, the dynamic program has to be solved directly and then an extensive Monte Carlo simulation study is performed to analyze the behavior of the optimal policy. The simulation results suggest several important insights regarding the timing and frequency of considering partial information in an IPD environment.

Keywords: Integrated product development, overlapping, information exchange, dynamic programming

1. Introduction

Integrated Product Development (IPD) advocates overlapping practices through enhanced interaction between activities of a product development process as a means to improve development performance (e.g., time and cost); see, for example, Gerwin and Barrowman (2002). Overlapping entails the partial or complete parallel execution of nominally sequential activities through the exchange of partial (i.e., incomplete or non-finalized) development information (Krishnan et al., 1997). Since overlapping and interaction increase the need for coordination among activities, IPD compensates by using organizational mechanisms (such as cross-functional teams) and technical mechanisms (such as Product Information Management (PIM)/Product Data Management (PDM) tools). PIM/PDM software is successful at managing the access and control of finalized information; however, they fail to handle the evolving nature of incomplete information that characterizes IPD environments, particularly at the early stages of development (Liu and Xu, 2001; Banker et al., 2006; Bardhan, 2007). The influence of organizational mechanisms on integration and coordination have been studied extensively in the product development literature (Hauptman and Hirji, 1999; McDonough, 2000; Hoegl et al., 2004); however, rigorous analysis concerning technical mechanisms within a development environment is scarce, despite the consensus on its pivotal role (Nambisan, 2003, 2009; Yassine et al., 2004; Banker et al., 2006).

In IPD environments, individual decisions are not made in isolation but are impacted by information generated and consumed by other development participants. When an individual participant reacts to newly arrived information, she will modify the requisite information for other dependent participants. This will create a complex chain of interdependencies, where the decision of a single participant has the potential to propagate throughout the development organization involving many other participants (Yassine et al., 2003). We specifically consider the archetypal scenario where a development participant is capable of accessing, at any time, unreliable, but related, development information, which has the potential to change as the development endeavor progresses. The participant has to decide what the appropriate action should be in response to this information. For instance, she can choose to ignore the information and continue with her original mission or she can incorporate the information to modify her work appropriately in light of this new, but partial, information. The trade-off involved here is that acting upon such
information may improve the quality of her work; however, there is a risk of disrupting her progress (to check this partial information) and then discovering that this newly available information is irrelevant.

In this article, we first consider the information exchange occurring during an IPD process composed of two nominally sequential activities: an upstream activity (A) and a downstream activity (B). The upstream activity feeds the downstream activity with information. The downstream activity has two options: (i) either abide by the sequential dependency and wait for the upstream activity to finalize its information before utilizing the predecessor information (Fig. 1(a)) or (ii) request that it receives incomplete, partial information intermittently from the upstream activity, which can be utilized, with caution, earlier than the scheduled final release of this information (Fig. 1(b)). In this article, we assume that the downstream activity selects option (ii).

This simple, but interesting, trade-off between time and uncertainty is the hallmark of most, if not all, overlapping models in the product development literature. We investigate this trade-off by developing a dynamic programming model that minimizes overall IPD completion time by controlling the amount of rework performed, taking into consideration the quality of partial information exchanged and the time for performing rework. In Sections 3 and 4, we show the existence of an optimal partial information exchange policy and derive simple rules to help the downstream decision maker (i.e., activity B) in deciding whether to consider the upstream information or to ignore it during overlapping. It is worth noting that we take the perspective of the downstream activity (B) as it is the dependent activity and although its decisions impact overall development time, it does not impact activity (A). Furthermore, we assume that the overlap time starts when the upstream activity (A) is able to provide information to downstream activity (B), which is known a priori. The downstream activity checks whether the upstream information is useful at every time step, starting from the beginning time of the upstream activity. However, no upstream information will be available (to communicate downstream) until some amount of time elapses on the upstream activity; activity (A) has to perform some work first (Yassine et al., 2008). Additionally, the overlapping time ends when the upstream activity (A) is completed, since the relation between both activities is finish-to-start. In the extreme scenario, both activities start together. In this case, the upstream activity (A) feeds the downstream activity (B) until the completion of activity (A). If the nominal duration of activity (A) is shorter than that of (B), then the last information sent from (A) to (B) is the end of the overlapping time, where activity (B) receives complete and perfect information. If the nominal duration of (A) is longer than that of (B), then activity (B) has to wait until the completion of activity (A) to perform the required rework because of the finish-to-start relation. In this article, we are investigating the dynamics of information flow between the upstream and downstream activities during this overlap period and simultaneously finding the optimal overlapping magnitude.

Then, in Section 5, we consider a development process composed of three nominally sequential activities, as shown in Fig. 2(a): an upstream activity (A), a midstream activity (B), and a downstream activity (C). In this case, the upstream activity is an independent activity that does not need any input or information from any other activity, the midstream activity is dependent on the upstream activity, and the downstream activity is dependent on the midstream activity. The relations between all activities are finish-to-start and thus the same assumptions made in the case of two-activity processes regarding their overlapping
structure are adopted here for the upstream, midstream, and downstream activities (see Fig. 2(b)).

We model this new process by developing an integrated dynamic programming model where the midstream activity (B) makes the decision of considering the information or not in coordination with both activities (A) and (C). In this model, the midstream activity (which is again the focal point of the proposed model) coordinates with the downstream activity an information exchange policy that minimizes rework for the whole IPD process (the midstream and downstream activities in this case) regardless of its locally optimal policy (i.e., when the downstream activity is ignored). In Section 6, numerical results are presented using simulation to compare the behavior of the midstream activity using the three-activity integrated model versus its behavior using the two-activity model (which we call the isolated model) and draw insights about information exchange policies to emphasize the importance of coordination among activities in IPD environments for rework reduction. The simulation results suggest that the integrative midstream activity tends to consider more partial information earlier compared to its locally optimal information exchange policy. Finally, we conclude by summarizing the work done and proposing potential research extensions in Section 7.

2. Literature review

Early pioneering research on the significance of overlapping strategy recommended that companies trying to implement concurrent engineering should stress the importance of sharing preliminary design information between participants at an early stage instead of late release of complete information (Clark and Fujimoto, 1991; Eisenhardt and Tabrizi, 1995). Additional empirical evidence from the global electronics industry statistically confirmed the effectiveness of overlapping development activities in reducing development lead times, particularly in organizations with fast uncertainty resolution (Terwiesch and Loch, 1999).

This empirical research provided support, motivation, and a launching pad for a stream of formal mathematical models to describe the overlapping development strategy and its proper management (Krishnan and Ulrich, 2001). The core trade-off in many of these models is to strike a balance between the time duration spent on working alone on a specific design activity versus the time spent communicating with other designers (Ha and Porteus, 1995; Krishnan et al., 1997; Loch and Terwiesch, 1998; Joglekar et al., 2001; Yassine et al., 2008; Lin et al., 2010). While communication reduces the chance of future rework, it prevents designers from spending time on actually performing their job. The objective function in all these models is to (i) determine the optimal amount of overlap between activities to reduce development lead time and/or (ii) determine the optimal communication policy (i.e., number of partial information exchanges between design activities) that reduces development lead time. In this article, we simultaneously derive the optimal amount of overlap between sequential activities and determine the optimal communication policy between the activities.

Ha and Porteus (1995) analyzed a development project that has a product design phase and a process design phase, which are executed in a parallel fashion. The purpose of the model was to find an optimal “progress review” strategy that minimizes the total development completion time. They considered the question of when to communicate partial information: that is, the meeting time. In contrary, in this article, we consider a more realistic scenario where we model whether or not to use a released piece of partial information.

Krishnan et al. (1997) developed an overlapping framework for two sequential development activities based on a downstream rework formulation that depends on upstream information evolution and downstream sensitivity. Based on these two constructs, they were able to find the optimal time to start overlapping. Again, in our model we are not merely concerned with when to start the downstream activity but we are also interested in the use of upstream information.

Loch and Terwiesch (1998) expanded this line of work to derive an optimal concurrency and communication policy for two overlapped activities to minimize development lead time. They modeled partial information as engineering changes that happen to upstream activities. These changes are then released to the downstream activity in batches. They accounted for communication and uncertainty effects in determining the optimal degree of concurrency and showed that improved communication allows increased overlapping. A further extension by Joglekar et al. (2001) accounted for resource constraints and a development deadline in a two-activity development scenario. Their model shows why different levels of concurrency (among development activities) can be optimal based on the discrepancy between “local” and “system” performance accrual rates.

More recent work of Yassine et al. (2008) considered a more realistic scenario where partial upstream information is always available to downstream activities, but the downstream activity has to make a decision on whether or not to use this information. The result of this scenario was a dynamic program formulation that provided a simple threshold policy for the downstream activity to guide its decision. In this current article, we use a similar decision for the downstream activity, but we additionally introduce a stochastic component that reflects the quality of received information.

Finally, Lin et al. (2010) formulated a mathematical model to investigate the time/cost trade-offs involved in a concurrent product development process to determine the optimal overlapping and communication strategies.
However, their model is deterministic and does not address the risk (i.e., the uncertainty) of the information exchanged between the upstream and downstream activities. In contrary, as mentioned earlier, our model accounts for such uncertainty, which makes it more realistic.

Most important, our model expands the literature on information exchange to formulate overlapping between three nominally sequential activities. This is the core difference between existing overlapping models in the literature (that are primarily concerned with information flows of incomplete information between an upstream and a downstream activity) and our proposed IPD model, which is additionally focused on incorporating downstream concerns. Our three-activity approach allows us to understand how the optimal information flow in an overlapped two-activity process would be augmented in the presence of downstream concerns.

3. The integrated two-activity process model with no fixed setup time

In this section, we develop a dynamic programming model to minimize the overall completion time of an IPD process composed of two activities: an upstream activity (A) and a downstream activity (B). Activity (B) can be executed sequentially after the completion of activity (A), in which case the overall duration of the development process becomes the sum of the nominal durations of both activities. Alternatively, as we assume in this article, in an integrated development environment, activity (B) can be executed concurrently with activity (A). We divide the duration of any activity into two parts: “nominal” activity duration and the “rework” activity duration. The nominal duration of an activity is the time needed for the activity to complete its assigned work assuming that it is either independent of all other activities (i.e., does not need information from other activities) or all of its requisite information is available at its start time. The rework duration is the extra time needed to perform rework in the case where the activity started with missing or incomplete requisite information. The nominal duration of an activity is assumed to be known and fixed; however, rework duration is stochastic and depends on the fraction of nominal work performed prior to the arrival of the incomplete information. Thus, the minimization of the completion time of the development process is equivalent to the minimization of the cumulative rework durations performed by the downstream activity.

We adopt a discrete-time framework with \( n \) time periods. The upstream activity continuously (i.e., at every time interval \( i = 1, \ldots, n \)) sends information to the downstream activity. The downstream activity has the ability to consider the information sent or to ignore it. At every time \( i \), the decision maker has two options: either not to consider the information at this time, which will lead to an accumulation of rework and extra rework at later time, or to consider the information at this time with a risk of rejecting it (i.e., not performing rework) due to low information quality. It is mandatory that at the last time interval (denoted by \( n \)) the decision maker considers the information in order to finalize the integration of the downstream activity with the upstream activity. This assumption is legitimate since the relation between both activities is of the finish-to-start type.

The earlier the downstream activity considers upstream information, the less rework is required. However, considering the information at early stages has a high probability of rejection since the quality of information might not be high enough to justify performing rework. We define \( p_i \) as the probability of not performing rework at time interval \( i \) due to the lack of upstream information quality. The value of the probability, \( p_i \), is a non-increasing function in time, \( i \) reaching \( p_n = 0 \) at the last time interval (i.e., completion of upstream activity (A)). That is, the quality of information improves with time. We assume that the probability \( p_i \) only reflects the quality of information transmitted from the upstream activity (A) at time \( i \). As such, this probability is independent of the state of the downstream activity (B). This is a reasonable assumption given that the information flows downstream only, and there is no feedback from (B) to (A). One may argue that \( p_i \) could also depend on the last time the downstream activity performs rework, \( r \). For example, \( p_i \) could be decreasing in \( r \) due to a synergistic combination of current and previously received information (reflected by \( r \)). However, this synergy of current and past information could be captured, in part, if not completely, by a proper calibration of the backlog rework function \( \gamma(i - r) \).

Additionally, delaying the evaluation of upstream information carries some penalty due to (i) the increase of nominal work complexity as the development evolves and (ii) the increase in the rework complexity of the unfinished work performed downstream without input from upstream, as delaying the consideration of information will increase the backlog of activities requiring input from upstream. In other words, the more we delay considering the information, the more we delay potential rework, and thus the more rework we have to perform due to increase of the complexity of both the nominal work and rework. Therefore, the potential rework at time \( i \) is a non-decreasing function of time \( i \) reflecting the added complexity of nominal work and of the amount of backlog, \( i - r \), where \( r \) is the last time rework is done. To reflect this, we express the potential rework as \( \alpha \gamma(i - r) \), where \( \alpha \) is non-decreasing in \( i \) and \( \gamma(i - r) \) is a non-decreasing convex function of \( i - r \) with \( \gamma(0) = 0 \). The convexity requirement is realistic as it implies that the rework complexity increases at an increasing rate in the backlog amount. This is needed in the following to derive realistic and practical threshold policies for information flow.

At each time period, \( i = 1, \ldots, n \), and depending on the amount of rework previously performed, as measured by the last time rework is done, \( r < i \), the decision maker chooses the optimal rework policy represented by the two
following options in order to minimize overall rework for the activity. The first option, not considering the information, will transfer the rework at time $i$ to the next time step. For the second option, considering the information, the amount of rework to be performed is equal to the combination of all of the cumulative work since last rework with probability $(1 - p_i)$ and is equal to the transfer of the cumulative work to the next time step with probability $p_i$. Figure 3 illustrates this decision situation and its corresponding sequence of events.

Therefore, the resultant Dynamic Programming (DP) formulation of this problem is represented by the following backward recursive equation:

$$R_i(r) = \min\{R_{i+1}(r), [\alpha_i \gamma(i-r) + R_{i+1}(i)](1 - p_i) + [R_{i+1}(r)]p_i\},$$

$$i = n - 1, \ldots, 1, \ r = 0, 1, \ldots, i - 1,$$

(1)

where $R_i(r)$ is the minimum expected rework to be performed at times $i, i + 1, \ldots, n$, when the last time rework was done is $r < i$. In Equation (1), the state of the DP is represented by the last time rework is done, $r$, with $r = 0$, indicating that no rework has been done yet. The formulation of Equation (1) reflects the decision tree in Fig. 3 and the fact that when rework is done at time $i$, the new system state is $i$. The boundary conditions for this DP are as follows:

$$R_n(r) = \alpha_n \gamma(n-r), \ r = 0, 1, \ldots n - 1.$$

The DP (1) indicates that information should be considered at time $i$ when the last time rework is done at $r$, if and only if

$$R_{i+1}(r) > \alpha_i \gamma(i-r) + R_{i+1}(i).$$

(2)

Analyzing Equation (2), we show that it always holds. That is, a dominant policy exists, which is to consider the information at every time period $i$ regardless of the probability of not performing rework due to the low quality of the information, $p_i$. Theorem 1 formalizes this result.

**Theorem 1.** In the absence of fixed setup time for rework and of considering the information, the optimal policy is to consider the information at every time period.

**Proof.** See Appendix A. 

Without the fundamental insight in Theorem 1, one may take for granted that the trade-off related to the extra rework due to delay in considering the information (reflected by increasing rework rates, $\alpha_i$ and $\gamma(i-r)$) along with the improvement of information over time (reflected by decreasing probabilities $p_i$) creates a threshold policy for considering the information. However, the DP formulation confirms the existence of a single dominant policy in this case, which is to always consider the information at every time interval regardless of the quality of this information. We relate this somewhat surprising finding to the absence of fixed setup time for processing the information and performing rework. That is, when information is free, it is advisable to consider it regardless of its quality. Indeed, when fixed costs are introduced into the model, a threshold policy exists, as shown in the next section.

4. The integrated two-activity process model with fixed setup time

The DP formulation in Section 3 has neglected the fixed setup time for considering the information, denoted by $f$, and of performing rework, denoted by $\beta$, in each time interval $i$ for the downstream activity (B). In this section, these two parameters are included in the DP formulation, which yield a new model, having a similar objective and decision variables as that in Section 3, shown in Equation (3) with the boundary condition given in Equation (4):

$$R_i(r) = \min\{R_{i+1}(r), [\alpha_i \gamma(i-r) + \beta + R_{i+1}(i)](1 - p_i) + [R_{i+1}(r)]p_i + f\},$$

$$i = n - 1, \ldots, 1, \ r = 0, 1, \ldots, i - 1.$$

(3)

$$R_n(r) = \alpha_n \gamma(n-r) + \beta + f, \ r = 0, 1, \ldots, n - 1.$$

(4)
The decision situation and corresponding sequence of events for the DP model in Equation (3) is similar to that of the model in Section 3 with the addition of $f$ and $\beta$, as shown in Fig. 4. Next, we analyze the DP in Equation (3) in order to develop an optimal information exchange policy. The following theorem develops important structural properties of the DP value function toward this end.

**Theorem 2.** The value function $R_i(r)$ is a concave decreasing function in $r$ with a slope

$$\frac{\partial R_i(r)}{\partial r} = -\alpha_i \frac{\partial \gamma(i-r)}{\partial (i-r)}.$$

**Proof.** See Appendix A.

**Theorem 3.** In the presence of fixed setup time for rework and considering the information:

(i) if $\hat{R}_i(r) = \alpha_i \gamma(i-r) + \beta + R_{i+1}(i) + (f/(1-p_i))$, then never consider the information at time $i$;

(ii) otherwise, there exists a threshold state value $r_i^*$ such that it is optimal to consider the information if and only if $r \leq r_i^*$ at time $i$.

**Proof.** See Appendix A.

The condition in Theorem 3(i) can be explained by defining $\hat{R}_i(r) = \alpha_i \gamma(i-r) + \beta + R_{i+1}(i) + (f/(1-p_i))$ as the expected rework between times $i$ and $n$ in state $r$ conditional on performing rework at time $i$. To understand this definition, let $I_i = 1$ if rework is performed at $i$ and $I_i = 0$ otherwise, and let $\hat{R}_i(r) = (1-p_i)[\alpha_i \gamma(i-r) + \beta + R_{i+1}(i)] + f$ be the expected rework between $i$ and $n$ when rework is performed at $i$. Then, $\hat{R}_i(r) = \hat{R}_i(r)/P[I_i = 1]$. It can be easily seen that information should be considered at time $i$ if the total expected value of rework given that rework is performed at $i$ is larger than that if the information is not considered at $i$, i.e., $\hat{R}_i(r) < R_{i+1}(r)$. The condition in Theorem 3(ii) states that if it is optimal not to consider the information at time $i$ when no rework has been done before ($r = 0$), i.e., if $R_{i+1}(0) < \hat{R}_i(0)$, then the dominating decision policy is not to consider the information in any other state (with $r > 0$) at $i$. This result is intuitive and would hold when the fixed costs of processing the information and rework, $f$ and $\beta$, are high. Theorem 3(ii) also indicates that when the dominance condition, $R_{i+1}(0) < \hat{R}_i(0)$, does not hold, i.e., when the fixed setup time are not too high, then there exists a threshold state value $r_i^*$ such that it is optimal to consider the information if the last time rework was done is below or equal to $r_i^*$, $r \leq r_i^*$. Figure 5 illustrates the two decision scenarios considered in Theorem 3 given the structural properties of the value function in Theorem 2. Note finally that comparison with Theorem 1 of the previous section confirms that the main reason for the existing of the threshold policy is the fixed setup time; i.e., one considers the information when the expected savings brought by the potential rework exceed the incurred fixed setup time.

Theorem 3 allows determining the decision policy at each decision epoch $i$ in terms of never considering the information or developing a threshold policy via $r_i^*$ easily with minimum storage requirement. An algorithm for doing this can be found in Appendix B.

As an example, consider a development process for an improved laptop table, consisting of two stages: an upstream marketing research stage (activity (A)) that feeds into a downstream engineering design stage (activity (B)). The duration of the upstream stage is 6 weeks and the duration of the design stage is 12 weeks. Marketing research will be ready to share preliminary marketing information about customer needs and desires after 2 weeks (i.e., after completing 50% of customer interviews). Marketing will also
have updated information ready at the end of each week thereafter. For the design stage, the setup time to consider the information \( f \) is 0.1 weeks (i.e., half a day), which is primarily the time consumed in reviewing the marketing update and discussion of these results with the engineering team. The rework setup time \( \beta \) is 0.2 weeks (i.e., 1 day), which is primarily the time consumed in locating and preparing the required CAD models. Finally, the probability of not performing rework \( p_i \) is a linear function that starts at one at time \( i = 0 \) and goes to zero at \( i = n \), which is a realistic assumption as it implies that no rework will be performed at time 0 and rework will definitely be performed at time \( n \). The potential rework function \( \alpha_i(i-r) \) is assumed to be \( \alpha_i(i-r), \) where \( \alpha_i \) is non-decreasing in \( i \) and \( \gamma(i-r) \) is linear as depicted in Table 1. Applying Theorem 3 for this IPD scenario to determine whether and when the design team should consider marketing information yields the last row in Table 1. The threshold policy in Table 1 indicates that design team should never consider preliminary marketing information before time week 3. In week 3, the design team should consider the marketing information. In week 4, the threshold policy indicates that if rework was performed in week 3 (with 30% probability), then marketing information should be considered, and so on.

5. The integrated three-activity process model

In this section, we consider an IPD process composed of three activities: an upstream activity (A), a midstream activity (B), and a downstream activity (C). All of the parameters used to govern the relationship between activities (A) and (B) in Section 4 remain the same for the relationship between (B) and (C). However, in this new relationship between (B) and (C), the probability \( p_i \) of not performing rework for activity (C) is a function of the quality of information received from (B). That is, the more (B) does rework based on information received from (A), the better information is sent to (C) via (B). For simplicity, we assume that information flows from activity (A) to (B) and from activity (B) to (C) at the same time instant \( i \) (i.e., information flows occur simultaneously) as shown in Fig. 2(b).

Specifically, we denote \( p_{i}^{C}(r_{B}) \) as the probability of activity (C) not performing rework at time \( i \), which is a function of the last time activity (B) has performed rework, \( r_{B} \), based on the information received from activity (A). We assume that \( p_{i}^{C}(r_{B}) \) is non-increasing in \( r_{B} \) reaching a minimum of \( p_{i}^{C}(i) \), which is realized when (B) performs rework at time \( i \) and (C) instantaneously considers the information. This assumption implies that the expected rework for (C) decreases as (B) performs more rework. Note that (C) can be seen as the downstream activity in a two-activity, (B) and (C), model similar to the model in Section 4. For that model, it can be easily shown that the value function for the downstream activity in (6), \( R_{i}(r) \), is decreasing in \( p_{i} \), which formally proves that the decrease of \( p_{i}^{C} \) via the rework of (B) benefits (C).

Next, we present our integrative DP model. This model gives the expected rework for both (B) and (C) with a two-dimensional state variable given by the last times these

Table 1. Example input/output data

<table>
<thead>
<tr>
<th>Period, ( i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_{i} )</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>( p_{i} )</td>
<td>0.9</td>
<td>0.8</td>
<td>0.7</td>
<td>0.6</td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>( r_{i}^{*} )</td>
<td>Never consider</td>
<td>Never consider</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>Consider</td>
</tr>
</tbody>
</table>
activities performed rework, \( r_B \) and \( r_C \). A similar notation to the two-activity model in Section 4 is utilized here but with superscripts B and C indicating the parameters related to activities (B) and (C), respectively. There are four possible cases at every time step \( i \). In the first case, both activities (i.e., B and C) do not consider the information. The second case is when activity (B) considers the information and activity (C) does not consider the information. In the third case, activity (B) does not consider the information but (C) considers the information. The fourth case is when both activities consider information simultaneously and (C) realizes an immediate potential benefit via a low probability of not performing rework, \( p_C^C(i) \). The corresponding DP formulation is shown in Equation (5) with the boundary conditions shown in Equation (6):

\[
R_i(r_B, r_C) = \min \left\{ R_{i+1}(r_B, r_C), \left[ \alpha_i^B \gamma^B(i - r_B) + \beta^B + R_{i+1}(r_B, r_C) \right] p_i^B + f^B, [\alpha_i^C \gamma^C(i - r_C) + \beta^C + R_{i+1}(r_B, i)] (1 - p_i^B) + [R_{i+1}(r_C)] p_i^C(1 - p_i^B) + [\alpha_i^C \gamma^C(i - r_C) + \beta^C + R_{i+1}(r_B, i)] (1 - p_i^B) + \left[ R_{i+1}(r_C) \right] p_i^C(1 - p_i^B) \right\},
\]

\( i = n - 1, \ldots, 1, r_B = 0, 1, \ldots, i - 1, r_B = 0, 1, \ldots, i - 1 \).

\[
R_i(r_B, r_C) = \alpha_i^B \gamma^B(n - r_B) + \beta^B + f^B, + \alpha_i^C \gamma^C(n - r_C) + \beta^C + f^C, r_B = 0, 1, \ldots, n - 1, r_C = 0, 1, \ldots, n - 1.
\]

The decision situation and corresponding sequence of events for the DP model in Equation (5) are as shown in Fig. 6. To analyze Equation (5), we need to consider two-dimensional states with four values to compare. Thus, it is difficult to derive threshold policies in this case as we did for the two-activity IPD process in Section 4. Therefore, we resorted to directly solving the DP recursion, Equation (5), and storing all of the decisions (i.e., value functions \( R_i(r_B, r_C), i = 1, \ldots, n - 1, r_B = 1, \ldots, i - 1, r_C = 1, \ldots, i - 1 \)) for all possible states. Although this direct solution

![Fig. 6. Decision tree for the integrated three-activity model.](image-url)
approach may require some storage, it is more efficient than other approaches such as attempting to solve Equation (5) by enumerating all possible policies via Monte Carlo simulation. The simulation results that we report in the next section are only helpful in validating a policy and are not used to create one.

6. Numerical results and analysis

In this section, we investigate the behavior of the three-activity model various scenarios using Monte Carlo simulation. We simulate the interaction between activity (B) and activity (C) by generating information qualities according to the probabilities \( p_i^B \) and \( p_i^C \), when needed, and using the values \( R_i(r_B, r_C) \) obtained from solving the DP Equation (5) to decide on considering the information or not. The analysis is focused on comparing the effect of integration on the frequency and timing of considering the information by the midstream activity (B) through comparison to the two-activity model in Section 4. Accordingly, we present some important insights in an IPD environment.

In order to facilitate our analysis, we introduce two metrics: percentage variation of information (PVI) due to integration and the average time for considering information (\( T \)). Equation (7) shows the PVI expression, where \( m \) is the number of simulation runs (we use 100,000 runs, which yields tight confidence intervals on PVI). \( I_{mk}^{iso} = 1 \) if (B) considers the information at time period \( i \) in simulation run \( k \) of the integrated model, and \( I_{mk}^{int} = 0 \) otherwise, and \( I_{mk}^{iso} \) is a similar indicator variable for the “isolated” two-activity model consisting of the same (A) and (B) activities as the integrated model. In the simulation, \( I_{mk}^{iso} \) is determined based on an exhaustive a priori storage of the values of \( R_i(r_B, r_C) \), while \( I_{mk}^{int} \) is determined based on the threshold policy of Theorem 3. If the PVI is positive, then (B) considers more information in the integrated model compared to the isolated model:

\[
PVI = \frac{\sum_{k=1}^{m} \sum_{i=1}^{n-1} (I_{mk}^{iso} - I_{mk}^{int})}{\sum_{k=1}^{m} \sum_{i=1}^{n-1} I_{mk}^{iso}}.
\]

Furthermore, we utilize metric \( \bar{T} \) (shown in Equations (8) and (9)) to compare the average time the information is considered by activity (B) between the isolated and the integrated case (we borrowed the concept of “duration” \( \bar{T} \) from finance, where it is used to denote the average timing of cash flows; Lundgren (1998)). The duration is a weighted average of the times of information exchange (i.e., \( 1, 2, \ldots, n - 1 \)), with the weight being the frequency of (B) considering the information at each time in our Monte Carlo simulation. That is, a duration of 4.5 means that on average (B) considers the information at time 4.5. This duration captures both the timing and intensity of information processing by (B). The smaller the duration, the earlier in the time horizon (B) is considering the information. If \( \bar{T}^{iso} \) is less than \( \bar{T}^{int} \) then (B) considers information earlier in the integrated case compared to the isolated case:

\[
\bar{T}^{iso} = \frac{\sum_{i=1}^{n} \sum_{k=1}^{m} I_{mk}^{iso}}{\sum_{i=1}^{n} \sum_{k=1}^{m} I_{mk}^{iso}}.
\]

\[
\bar{T}^{int} = \frac{\sum_{i=1}^{n} \sum_{k=1}^{m} I_{mk}^{int}}{\sum_{i=1}^{n} \sum_{k=1}^{m} I_{mk}^{int}}.
\]

As a follow-up to our earlier example in Section 4, we introduce a third stage (activity (C)), which is the process design stage. Then, the base IPD process has 10 weeks (\( n = 10 \)) and the following data. For activity (B), the setup times for considering the information and for performing rework are \( f_B = 0.1 \) and \( \beta_B = 0.2 \), respectively. These values are also \( f_C = 0.1 \) and \( \beta_C = 0.2 \) for activity (C). Both activities ((B) and (C)) have the same “nominal” rework rates \( \alpha_i \) and quality of information probabilities \( p_i \), which are shown in Table 1. The “backlog” rework functions are assumed to be linear for both (B) and (C); i.e., \( \gamma_B(i - r_B) = i - r_B \) and \( \gamma_C(i - r_C) = i - r_C \). For simplicity, we assume that at a given time \( i \), \( p_i^C(r_B) \) is the same for all \( r_B \) except for time \( i \), where \( p_i^C(i) = 0 \). Here, \( p_i^C(r_B) \) is non-increasing in \( r_B \) for \( r_B = 0, 1, 2, \ldots, i \). This non-increasing form of \( p_i^C(r_B) \) provides motivation for (B) to consider more information in an IPD environment and captures our key modeling ingredient in the integrated model, which is more rework of (B) implies better information to (C). This impulse structure encourages concurrent processing of information and rework by (C) (i.e., at the same time \( i \)). In addition, this impulse effect of rework does not seem to be restrictive. In our ongoing research, we are experimenting with smoother forms of \( p_i^C(r_B) \).

We have simulated an extensive number of cases for both models, but we choose to illustrate the few cases that show some major insights. Cases 1 to 7 are meant to draw insights regarding the information exchange policies. Table 2 is a summary of the numerical results with descriptions for every case. The following major insights are observed from Table 2.

Insight 1: Frequency of considering information: Activity (B) in the integrated model tends to consider more information than in isolation.

In the isolated model, activity (B) is not concerned with activity (C) and its behavior is to minimize its own expected rework. In the integrated model, the behavior of (B) changes to help (C) reduces its rework by considering more information. Figure 7 shows that in the integrated model and in all the cases in Table 2, activity (B) considers information at least the same number of times as in the sequential model, as reflected by the PVI values in this figure.

Insight 2: Timing of considering information: Activity (B) in the integrated model tends to consider the information earlier than in isolation.
Table 2. Simulated case scenarios

<table>
<thead>
<tr>
<th>Case</th>
<th>Change to base case</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>$\beta^C = 0$ and $f^C = 0$</td>
<td>(B) starts considering the information at a higher frequency ($PVI = 19.66%$) and at an earlier time ($\bar{T}<em>{iso} - \bar{T}</em>{int} = 0.5281$) in the integrated model</td>
</tr>
<tr>
<td>Case 2</td>
<td>$\beta^C = 0.5$ and $f^C = 0.5$</td>
<td>(B) considers information at a higher frequency ($PVI = 4.32%$) and at an earlier time ($\bar{T}<em>{iso} - \bar{T}</em>{int} = 0.02906$) in the integrated model</td>
</tr>
<tr>
<td>Case 3</td>
<td>$\beta^C = 1$ and $f^C = 1$</td>
<td>(B) considers information at a slightly higher frequency ($PVI = 0.68%$) and at an earlier time ($\bar{T}<em>{iso} - \bar{T}</em>{int} = 0.00212$) in the integrated model</td>
</tr>
<tr>
<td>Case 4</td>
<td>$\beta^C = 1.1$ and $f^C = 1.1$</td>
<td>(B) follows the same policy as in the isolated model because (C) has high fixed setup time (i.e., $PVI = 0%$ and ($\bar{T}<em>{iso} - \bar{T}</em>{int}$) = 0)</td>
</tr>
<tr>
<td>Case 5</td>
<td>$\beta^B = 0$ and $f^B = 0$</td>
<td>(B) has no fixed setup time, thus it always considers the information in both cases (i.e., isolated and integrated; i.e., $PVI = 0%$ and ($\bar{T}<em>{iso} - \bar{T}</em>{int}$) = 0)</td>
</tr>
<tr>
<td>Case 6</td>
<td>$\beta^B = 0.2$ and $f^B = 0.2$</td>
<td>(B) has medium fixed setup time. Thus, in isolation it does not consider much information. However, when integrated with (C), it considers much more information and early on (i.e., $PVI = 16.03%$ and ($\bar{T}<em>{iso} - \bar{T}</em>{int}$) = 0.467)</td>
</tr>
<tr>
<td>Case 7</td>
<td>$\beta^B = 0.5$ and $f^B = 0.5$</td>
<td>(B) never considers the information in both isolated and the integrated models because it has high fixed setup time (i.e., $PVI = 0%$ and ($\bar{T}<em>{iso} - \bar{T}</em>{int}$) = 0)</td>
</tr>
</tbody>
</table>

Since (C) must consider the information at the terminal time period $n$, then the “help” of (B) by improving the quality of information will be mostly needed in earlier time periods. Accordingly, we expect that (B) considers the information at earlier time periods in the integrated model. Figure 8 confirms this insight by showing the difference between the number of times activity (B) considers the information in the integrated model versus that for the isolated model for the seven cases in Table 2. Figure 8 displays the difference between the average time the information is considered by (B) between the isolated (two-activity model) and the integrated case (three-activity model), ($\bar{T}_{iso} - \bar{T}_{int}$). The bigger the difference, the earlier (B) considers the information under integration. Figure 8 clearly shows that when (B) considers more information as a result of integration—i.e., cases with high $PVI$ (e.g., the bases case, and cases 1 and 2)—it does so early-on.

Insight 3: Delay of considering the information: If fixed setup times for considering the information and for rework for either activity (B) or (C) are high, then integration is costly and prohibitive, which implies that (B) behaves the same in integration and isolation.

When $\beta^C$ and $f^C$ are large enough, the improvement of information quality by (B) will not benefit (C). That is, when $\beta^C$ and $f^C$ increase in cases 2 to 4, $PVI$ decreases as indicated in Fig. 7. This is because in these cases integration becomes costly and midstream activity (B) does not share more information under integration. A similar observation is noted when the fixed setup times for (B) are large ($f^B$ and $\beta^B$); e.g., in case 7 in Table 2. A useful note here is that midstream activity (B) considers more information when

![Fig. 7. $PVI$ for activity (B) (color figure provided online).](color figure provided online)
Fig. 8. Difference between the average time information is considered by activity (B) between the isolated and the integrated models, \((\bar{T}_{iso} - \bar{T}_{int})\), for all cases (color figure provided online).

its fixed setup times are low (e.g., case 5 in Table 2), as in these cases (B) always processes the information in both isolated and integrated settings.

7. Summary and conclusions

The overarching goal of this article is to support IPD practices by advancing the theoretical foundations for the required integration support tools represented by various PIM systems (Liu and Xu, 2001; Banker et al., 2006). This was accomplished by introducing the concept of evolving information that is partial and incomplete but potentially useful to share within an IPD environment if managed properly.

In this article we investigated the information exchange in an IPD environment. We first developed a model to reflect the information exchange between two activities, upstream and downstream, without any fixed delays. We showed the existence of a dominant policy, which is to always consider the information in the absence of fixed setup times for rework and for considering the information. Then, we derived an optimal threshold policy to consider the information in the presence of fixed setup time for rework and for considering the information. Furthermore, in Section 5 we introduced a midstream activity to develop a model for the information exchange between three sequential activities and analyzed the effect of upstream and downstream information exchange on this activity. Important insights were drawn to help decision makers formulate work policy procedures to exchange information between several activities for the same process.

The three main parameters required for running both the two-activity and the three-activity models are \(p_i\) (probability of not performing rework at time \(i\)), \(\alpha_i\) (nominal rework rate at time \(i\)), and \(\gamma(i - r)\) (backlog rework function). Indeed, these parameters are not completely new to the product development literature. The accessibility of these parameters (particularly upstream evolution and downstream sensitivity) has been illustrated in many previous studies (e.g., Krishnan et al. (1997), Loch and Terwiesch (1998), Roemer et al. (2000), and Lin et al. (2009), to name a few). Many of these authors have reported examples and case studies estimating these parameters for use in their proposed mathematical models. In all of these studies, the main instrument for measuring these parameters was interviews with subject matter experts with experience in similar projects. However, it is worth noting that what is important here is not the exact values of these parameters but rather the overall trend or functional properties; that is, \(p_i\) is a non-increasing function in time \(i\), \(\alpha_i\) is an increasing function in time \(i\), and \(\gamma(i - r)\) is increasing and convex in the backlog \(i - r\). In general, we are not very interested in final answers, but ultimately interested in higher-level managerial insights regarding IPD management; namely, the frequency and timing for considering incomplete upstream information to incorporate downstream concerns.

The insights drawn from the three-activity model are generalizable to all IPD processes of any size and remain useful as guidelines for all midstream activities. For instance, in order to benefit from integration with downstream activities, midstream activities should consider more information and will do so earlier. Also, when the fixed delays are high, midstream activities will act as if they are in isolation. These insights have been recorded in the literature by...
many authors based on empirical observations (Clark and Fujimoto, 1991; Eisenhardt and Tabrizi, 1995; Gerwin and Barrowman, 2002); however, these insights were not analytically supported as illustrated in our three-activity model.

IPD managers equipped with these insights can prepare for the proper management of IPD since now they know what to expect with integration and when it will occur. That is, more information will be shared and earlier than usual as compared with overlapping, which has been studied extensively in the product development literature. Alternatively, if managers are executing an IPD process and they find that midstream activities are not sharing more information earlier than with overlapping only, then they should investigate whether this is caused by the high overhead—i.e., high fixed rework and processing information delays—or simply they are not executing an optimal IPD process.

In anticipation for this expected increase in information exchange and the augmented timing, compared with traditional overlapping practices, IPD managers must ensure the availability of technology (i.e., communication channels) and people’s time to accommodate this change.

Although this article is unique in analyzing three-activity development processes and deriving optimal information exchange policies, a natural extension is to accommodate a larger chain of sequential activities. Our proposed IDP approach can be easily extended to solve larger problem instances (i.e., larger than three activities). This can be done by assuming that all downstream activities are related by the same type of parameters. That is, once the probabilities \( p_i \) and the rework rates \( \alpha, \gamma(i = r) \) are estimated for the downstream activities, the IDP recursion equation can be calculated. We would need to store, for \( n \) time periods and \( k \) activities, \( \sum_{i=1}^{n-1} i^k \) integers to have a full decision policy. This implies that with today’s computing power, our IDP framework can handle up to a dozen sequential activities with a realistic time horizon of up to a year. However, as the number of activities increases this would make the IDP computations more complex and tedious, which may require the development of heuristics or metaheuristic procedures to deal with such computational complexity. This approach would make an interesting research extension to this article and merits further examination. Furthermore, another interesting extension to our proposed model is to include feedback between these activities. That is, when a downstream activity detects an infeasability with upstream information and sends the information back. This requires more fundamental changes to our model and is worthy of further investigation.

References


Appendices

Appendix A: Proof of analytical results

The following lemma is needed in the proof of Theorem 1.
Lemma A1. The slope of the value function, $\partial R_i(r)/\partial r$, satisfies

$$\frac{\partial R_i(r)}{\partial r} \leq -\alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)}$$

Proof. The result holds at time $n$, $t = n$, as $R_n(r) = \alpha_n y_n(n - r)$ and

$$\frac{\partial R_n(r)}{\partial r} = -\alpha_n \frac{\partial y(n - r)}{\partial (n - r)}.$$

By induction, we assume the result holds for $t = i + 1, i + 2, \ldots, n$, and we show it to hold for $t = i$. Note that:

$$\frac{\partial R_i(r)}{\partial r} \leq \max \left( \frac{\partial R_{i+1}(r)}{\partial r}, -(1 - p_i)\alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)} + p_i \frac{\partial R_{i+1}(r)}{\partial r} \right) \leq \max \left( -\alpha_{i+1} \frac{\partial \gamma(i - r)}{\partial (i - r)} - p_i \alpha_{i+1} \frac{\partial \gamma(i + 1 - r)}{\partial (i + 1 - r)} \right)$$

$$\leq \max \left( -\alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)} - (1 - p_i)\alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)} - p_i \alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)} \right) \leq -\alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)},$$

where the inequality before last follows from the monotonicity of $\alpha_i$ and the monotonicity and convexity of $\gamma(i - r)$.

Proof of Theorem 1. The result holds if

$$R_{i+1}(r) > [\alpha_i \gamma(i - r) + R_{i+1}(r)](1 - p_i) + R_{i+1}(r)p_i,$$

which is equivalent to

$$R_{i+1}(r) - R_{i+1}(i) > \alpha_i \gamma(i - r).$$

Let $f(r) = R_{i+1}(r) - \alpha_i \gamma(i - r)$ and $f'(r) = \partial f(r)/\partial r$.

Note that

$$f(r) - f(i) = R_{i+1}(r) - \alpha_i \gamma(i - r) - [R_{i+1}(i) - \alpha_i \gamma(0)]$$

$$= R_{i+1}(r) - R_{i+1}(i) - \alpha_i \gamma(i - r).$$

The mean value theorem implies that there exists $c, r < c < i$, such that

$$f(r) - f(i) = f'(c)(r - i) = (r - i) \left( \frac{\partial R_{i+1}(k)}{\partial k} \right)_{k=c} - \alpha_i \frac{\partial \gamma(i - k)}{\partial k} \left|_{k=c} \right.$$  

$$= (r - i) \left( \frac{\partial R_{i+1}(k)}{\partial k} \right)_{k=c} + \alpha_i \frac{\partial \gamma(i - k)}{\partial (i - k)} \left|_{k=c} \right).$$

Applying Lemma A1 completes the proof.

Proof of Theorem 2. The result holds at time $n$, $t = n$, as $R_n(r) = \alpha_n y_n(n - r) + \beta + f$ is concave decreasing in $r$ with slope $-\alpha_n(\partial \gamma(n - r)/\partial (n - r))$. By induction, we assume the result holds for $t = i + 1, i + 2, \ldots, n$, and we show it to hold for $t = i$. Then,

$$R_i(r) = \min \{ R_{i+1}(r), [\alpha_i \gamma(i - r) + \beta + R_{i+1}(i)](1 - p_i) + [R_{i+1}(r)p_i + f] \},$$

is concave in $r$ since the minimum of two concave functions is concave (Bazara et al. (1993) Lemma 2.1.2, p. 35) and the sum of two concave functions is also concave. To prove that $R_i(r)$ decreasing in $r$, consider:

$$R_i(r) = \min \{ R_{i+1}(r + 1), [\alpha_i \gamma(i - r + 1) + \beta + R_{i+1}(i)](1 - p_i) + [R_{i+1}(r + 1)p_i + f] \}.$$

Comparing $R_i(r)$ to $R_i(r + 1)$ term by term, with the inductive argument, implies that $R_i(r) \geq R_i(r + 1)$. Therefore, $R_i(r)$ is decreasing in $r$. Finally, the slope result:

$$\frac{\partial R_i(r)}{\partial r} \leq -\alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)},$$

also follows from the inductive argument similar to the proof of Lemma A1.

Proof of Theorem 3. From Equation (3), it is optimal to consider the information at time $i$ if $R_{i+1}(r) \geq \tilde{R}_i(r)$, where $\tilde{R}_i(r) = \alpha_i \gamma(i - r) + \beta + R_{i+1}(i) + (f/(1 - p_i))$. Consider two cases:

Case 1: $R_{i+1}(0) < \tilde{R}_i(0)$, (Fig. 5(a)). Then, since from Theorem 2:

$$\frac{\partial R_{i+1}(r)}{\partial r} \leq \frac{\partial \tilde{R}_i(r)}{\partial r} \leq -\alpha_i \frac{\partial \gamma(i - r)}{\partial (i - r)} \frac{\partial R_{i+1}(r)}{\partial r},$$

it follows that $R_{i+1}(r) < \tilde{R}_i(r)$, for all $r = 1, 2, \ldots, i - 1$. 

Case 2: $R_{i+1}(0) \geq \tilde{R}_i(0)$, (Fig. 5(b)). Note that for $r = i$, $\tilde{R}_i(i) = \beta + R_{i+1}(i) + (f/(1 - p_i)) > R_{i+1}(i)$. Since as shown in Case 1, $(\partial R_{i+1}(r)/\partial r) \leq (\partial \tilde{R}_i(r)/\partial r)$, and both $R_{i+1}(r)$ and $\tilde{R}_i(r)$ are decreasing in $r$ (from Theorem 2 for $R_{i+1}(r)$), this implies that over the range of values $r = 1, 2, \ldots, i - 1$, $R_{i+1}(r)$ and $\tilde{R}_i(r)$ have exactly one intersection. Therefore, there exists a unique $r^*_i \in \{1, 2, \ldots, i - 1\}$, such that $\tilde{R}_i(r) < R_{i+1}(r)$ for $r \leq r^*_i$. 


Appendix B: Algorithm for the decision policy based on Theorem 3

The following algorithm determines the decision policy at time, \( i, i = 1, 2, \ldots, n - 1 \). It only requires storing two arrays \( R_0(.) \) and \( R_1(.) \).

**Step 0:** For \( r = 0 \) to \( n - 1 \),
\[
R_1(r) = \alpha n \gamma (n - r) + \beta + f
\]

**Step 1:** Set \( i = n - 1 \)

**Step 2:**
- If \( R_1(0) < \tilde{R}_0(0) \equiv \alpha i \gamma (i) + \beta + R_1(i) + \frac{f}{1-p_i} \), then
  - Never consider the information at time \( i \)
- Else
  - For \( r = 0 \) to \( i - 1 \)
    - If \( R_1(r) \geq \tilde{R}_0(r) \equiv \alpha i \gamma (i - r) + \beta + R_1(i) + \frac{f}{1-p_i} \), then
      - Set \( r_i^* = r \)
    - For \( r = 0 \) to \( i - 1 \)
      - \( R_0(r) = \min \{ R_1(r), [\alpha i \gamma (i - r) + \beta + R_1(i)](1 - p_i) + [R_1(r)]p_i + f \} \)
  - Set \( R_0(0) = R_0(0) \)
  - Set \( i = i - 1 \)
- If \( i > 0 \)
  - Go to Step 2
- Else
  - Stop.

---

**Biographies**

Ali Yassine is a Professor of Engineering Management and the program coordinator at the American University of Beirut. Previously, he was an Assistant Professor at the Department of IESE at the University of Illinois at Urbana-Champaign (UIUC). Prior to joining UIUC, he was a research scientist at the MIT Center for Technology, Policy and Industrial Development and a project manager at Chrysler International Corporation. He has authored over 60 articles in refereed academic journals and conferences, covering product development, engineering design management, project management, concurrent engineering, systems and entrepreneurial engineering, and management of technological innovations.

Bacel Maddah is an Associate Professor in the Engineering Management graduate program at the American University of Beirut. He obtained a Ph.D. in Operations Research from Virginia Tech. His research interests are in decision making under uncertainty and its application to areas such as retail, logistics, services, and, more recently, oil and gas and finance. His research has been published in journals such as *Management Science*, *Naval Research Logistics*, *European Journal of Operations Research*, and *Journal of the Operational Research Society*, among others. He has practical experience working as a business analyst with United Airlines and Hannaford Bros. He has also consulted on economic feasibility and logistics for several projects in the Middle East. He spent one semester at Virginia Tech as a Visiting Scholar and has been invited to several international universities to present his research.

Nabil Nehme is a Ph.D. candidate in Civil Engineering with an emphasis on transportation and operations research at the American University of Beirut. He received a degree in Mechanical Engineering and Master in Engineering Management. His research interests are in the field of operations research and game theory at the strategic and operational levels, with the application to maritime transportation. He has authored and co-authored several journal publications and conference proceedings. He is a member of ASME and INFORMS.