Submitted to *Operations Research*
manuscript (Please, provide the mansucript number!)

A Hierarchical Framework for Organizing a Software Development Process

Foaad Iravani
Foster School of Business, University of Washington, Seattle, Washington 98195,
firavani@uw.edu

Sriram Dasu
Marshall School of Business, University of Southern California, Los Angeles, California 90089,
dasu@marshall.usc.edu

Reza Ahmadi
Anderson School of Management, University of California, Los Angeles, Los Angeles, California 90095,
rahmadi@anderson.ucla.edu

Every year, companies that produce commercial tax preparation software struggle with thousands of state and federal changes to tax laws and forms. Three competitors dominate the market with its short selling season, and release delays slash profits. Tax authorities issue updates August-December, and all changes must be processed and incorporated before year end. Systematic resource allocation and process management are crucial yet problematic due to the volume and complexity of changes, brief production timeframe, and feedback loops for bug resolution. A leading tax software provider asked us to formulate systematic approaches for managing process flow and staffing development stages with the goal of releasing the new version on time at minimum cost. To that end, we develop deterministic models that partition tax forms into development groups and determine staffing levels for each group. Partitioning forms into groups simplifies workflow management and staffing decisions. To provide a range of resource configurations, we develop two modeling approaches. Numerical experiments show that our models capture the salient features of the process and that our heuristics perform well. Implementing our models reduced company overtime hours by 31% and total resource costs by 13%.

Keywords: product development, software development, tax software, workforce management, resource allocation, grouping index, integer programming

1. Introduction

Consumer tax preparation applications comprise a profitable niche in software development, an industry which the U.S. Bureau of Labor Statistics projects to be the second-fastest growing in the American economy through 2020 (*Bureau of Labor Statistics* 2012). Growth specifically in tax preparation software has been bolstered by Internal Revenue Service (IRS) efforts to achieve an

80% electronic filing rate for major returns by 2013. Reflecting those efforts, 2011 sales of consumer tax programs grew by as much as 20% with 40 million taxpayers using them to file their returns (*Electronic Tax Administration Advisory Committee* 2011, *Department of Justice* 2011).

The idea for this project grew from conversations with a software engineer who was studying for an MBA while working for one of the nation's largest tax preparation software providers. The engineer characterized the market as fiercely competitive, subject to a short and fixed sales window, and committed to a product that becomes obsolete every year. Three companies dominate this high-pressure arena, racing one another each year to incorporate changes to laws and forms, test their new versions, and release bug-free products for the upcoming tax season. Obviously, early release confers an advantage on the tax software development company (henceforth TSDC) by helping it maximize market share. Thus, delays can result in significant losses.

The development process for tax software is complex and consists of multiples stages that incorporate thousands of revisions. Some changes are trivial while others command significant developer time. Adding to the challenge is the fact that state and federal authorities only begin announcing their changes in early August and continue doing so through December, while mid-December marks the start of the tax software sales season. Each stage in the development process contains built-in feedback loops that interrupt workflow in order to correct errors. To release the application on time and control development costs, therefore, TSDC must effectively manage the process and accurately determine staffing levels. However, in the course of analyzing and observing a development cycle, we noted that tasks were assigned on an ad hoc basis, and staffing levels were subject to the vagaries of individual power and influence. Not surprisingly, the firm's bottom line chronically suffered from significant overtime costs, and TSDC found it difficult to achieve a timely release.

Clearly, the large number of forms calls for a staffing plan driven by an effectively organized development effort that simplifies day-to-day operations management, improves coordination among different stages, and facilitates information flow. To that end, we propose a model that restructures the development effort by sorting tax forms into multiple groups and determines staffing levels throughout the process. In answer to the challenge of making the two decisions simultaneously,

we employ a hierarchical approach in which we first form the groups and then allocate sufficient resources (we use the terms *resource* and *employee* interchangeably) to ensure timely completion. The decision to create groups is supported by studies in software development (e.g., Brooks 1975, Cusumano 1997) that demonstrate benefits such as increased effectiveness and enhanced quality arising from the division of tasks into groups.

Although we develop a hierarchical planning framework to organize tax software development, our approach could also be applied to development processes in other highly-regulated industries that periodically revise a product, face tight deadlines, and involve processing requirements similar to those we encounter here (e.g., Ahmadi et al. 2001). Every year, for example, the Centers for Medicare & Medicaid Services publishes regulatory updates to payment rules, standard assessments, and resource utilization group and case mix calculations. Companies producing Medicare and Medicaid billing software must incorporate these changes before October as efficiently as they can. Aerospace, cellular communication, healthcare, and enterprise resource planning (ERP) face similar development challenges. Development teams in these disciplines typically work in parallel, sharing a common deadline to complete their design and development tasks. Thus, task assignment and resource allocation are widespread issues.

The remainder of the paper is organized as follows. In Section 2, we review the relevant literature. In Section 3, we describe the problem in detail, explain our modeling assumptions, and propose approximations to capture the effect of feedback loops on process completion time. We introduce our notation and formulate our models in Section 4, describe their solution procedures in Section 5, and investigate the performance of the hierarchical approach and the solution procedures using numerical experiments in Section 6. Section 7 explores the implementation of our models at TSDC. Section 8 concludes the paper with a summary of our research. An electronic companion to this paper is available as part of the online version that can be found at http://or.journal.informs.org/.

2. Literature Review

The development process for tax preparation software has elements of reentrant flow shops (Graves et al. 1983) and other product development projects. On one hand, the process is repetitive because

the software is produced each year and each version encompasses multiple jobs, all facing the same deadline. On the other hand, task requirements vary significantly each year and TSDC cannot just repeat the same process. Every version of the application, therefore, can be thought of as a project.

In addition to traditional project management techniques (Tavares 1998), several other approaches have been proposed for reducing the duration of product development projects by changing the sequence of development activities, overlapping activities, and changing the flow of information among developers (Krishnan et al. 1997, Smith and Eppinger 1997, Carrascosa et al. 1998, Loch and Terwiesch 1998, Roemer et al. 2000, Ahmadi et al. 2001, Roemer and Ahmadi 2004). In our setting, however, the sequence of activities is fairly straightforward, offering minimal opportunity for modification or increased overlap. Therefore, we seek to achieve the desired duration of the development process by creating groups to facilitate the flow of the process and then staffing the stages properly.

Partitioning the development effort and allocating tasks among groups is a common software industry practice that substantially influences project duration, software quality, development cost, and reusability. Cusumano (1997) and Cusumano et al. (2003) survey techniques employed by leading software companies to partition and manage large projects. Such studies explore ways to divide complex tasks into manageable modules (Shaw and Clements 2006). In our problem, we use an index for grouping forms based on similarities between the amount of work they require. We then staff the groups to ensure timely completion.

In software engineering, an interesting dilemma is whether to assign one or two developers to work on a module. Empirical evidence suggests that assigning multiple developers increases the time and effort needed to develop a module but decreases the time and effort needed to integrate it. Dawande et al. (2008) develop a mathematical model to find conditions under which one approach supersedes the other. We assume that jobs can be divided among the developers, which is an approximation for tractability.

Browning and Ramasesh (2007) provide a comprehensive review of network-based process models for managing product development activities, and they cite very few papers that are concerned

with resource allocation. Joglekar and Ford (2005) study ways to dynamically shift a finite pool of resources across different stages of the process, using a procedure that is considerably simpler than ours. Though we are not concerned with dynamically reassigning resources, Joglekar and Ford intriguingly suggest that complexity diminishes the value of dynamic resource allocation.

In software development, another class of resource allocation problems addresses the optimal allocation of resources among competing priorities. Ji et al. (2005) explore optimal allocation between software construction and debugging to maximize quality. Kumar et al. (2006) look at the tradeoff between the benefits of adding a new feature and the risk of introducing new bugs with it.

Queuing network models also have been proposed for staffing projects. Adler et al. (1992) consider multiple product development projects that proceed through nodes representing departments or functional capabilities. Queuing models assume that the design facility is continually receiving design projects (Adler et al. 1992) or maintenance requests (Asundi and Sarkar 2005, Kulkarni et al. 2009, Feng et al. 2006), each with its own deadline. Although we have multiple tasks in the process, they are all part of a single project.

Resource scheduling problems also arise during software execution (Hos and Shin 1997) where the challenge is to complete a job on time while allocating the elements of the job in real time to a set of resources. Complexity in these problems stems from the nature of the precedence relationships and interdependencies among tasks. In our problem, the flow patterns and precedence relationships are simple. The challenge stems from the large number of tasks that need to be performed.

Kekre et al. (2009) address an interesting problem with features similar to our work. They analyze the workforce management of multistage check-clearing operations at a commercial bank. They use simulation–optimization to capture the tradeoff between efficiency and the risk of delayed checks resulting from excessive workforce reduction. Our problem involves staffing the stages of a process and deciding to which group each form should be assigned. The very large number of forms produces a correspondingly large number of scenarios for assigning forms to groups and allocating resources, even for three to five groups. Therefore, simulation–optimization is not well-suited to

solving practical instances of our problem, though one could use the technique to explore different staffing scenarios for a fixed assignment of forms to groups.

3. Problem Characteristics and Modeling Assumptions

Maintaining a tax preparation application encompasses multiple processes throughout the year. Some processes, such as maintaining the software engine at the core of the application, proceed independently of revisions to the tax code. Others are concentrated into the quarter before the impending tax season. These processes arise from the IRS and each state taxing authority independently constructing tax laws and forms. The number of changes is immense; they are released dynamically, starting in August and continuing into the tax software selling season; and their implementation encompasses highly variable degrees of difficulty. Five major stages dominate the workflow (see Figure 1).

1. The Image Development Group (IDG) evaluates all tax form and document changes and creates an image of each form.
2. Calculation (CALC) elucidates and tests the computations that underlie each form. CALC is the most important stage and carries the most amount of work.
3. Electronic Filing (EF) develops electronic versions of the forms, employing functions and macros based on the structure of each form and the fields it contains.
4. The Interview team designs the user interface that guides the consumer through the software.
5. Integration and Final Test (I&FT) incorporate the forms into the application and put each component through exhaustive trials. Integration also designs the buttons, menus, and toolbars. Final Test sends each error back to the team that introduced it.

The backward arcs in Figure 1 indicate feedback from I&FT to earlier, upstream stages. When an error is found in a form during I&FT, it is returned to the appropriate stage—CALC, EF, or Interview—for correction. Each stage, furthermore, consists of two substages, the first for processing forms and the second for testing the forms internally. The internal tests may return a form to their corresponding process for rework. Taking into account the number of changes to be incorporated

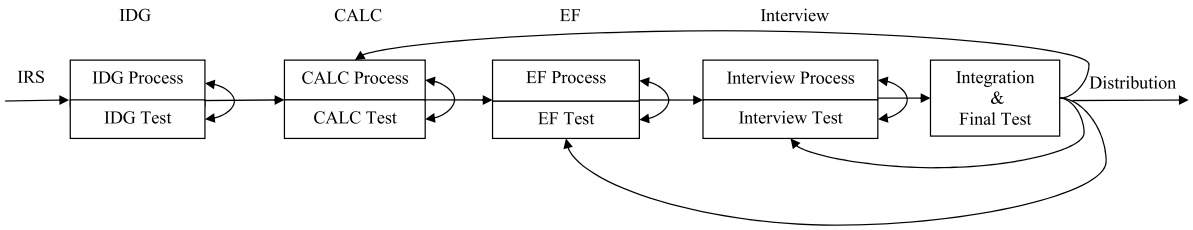

Figure 1 Tax software process line

and their dynamic introduction, workflow management and coordination becomes a formidable task.

3.1. Modeling Assumptions

To effectively control this development effort, we need to sort the forms into manageable groups, staff each group, and develop rules for sequencing the tasks. The work volume, dynamic arrivals, feedback loops, and processing time variability make it nearly impossible to identify good sequencing rules. Hence, we develop models that support tactical decisions about grouping and staffing, and we ignore operational issues such as sequencing and scheduling.

We make two simplifications while developing the models, one based on work forecasts and the other on downstream stage idle times during rework. Having observed that the system as a whole is never idle due to lack of work, we disregard pattern details associated with forms arriving dynamically. Instead, we base grouping and staffing decisions on work forecasts. If the forecast were to change significantly during the season, TSDC could always revisit the models and alter staffing levels. TSDC managers who participated in our study, validated this simplification based on their past experience, and we obtained objective validation via our computational experiments.

The second simplification concerns feedback loops, which increase the amount of work at each stage, introduce additional uncertainty into processing time requirements, and may temporarily put a stage out of action while a stage downstream creates a rework loop. Given our interest in completion time, the latter impact is the most problematic. The likelihood of inserted idle times depends on the initial amount of work; when the amount of work is high, inserted idle times are

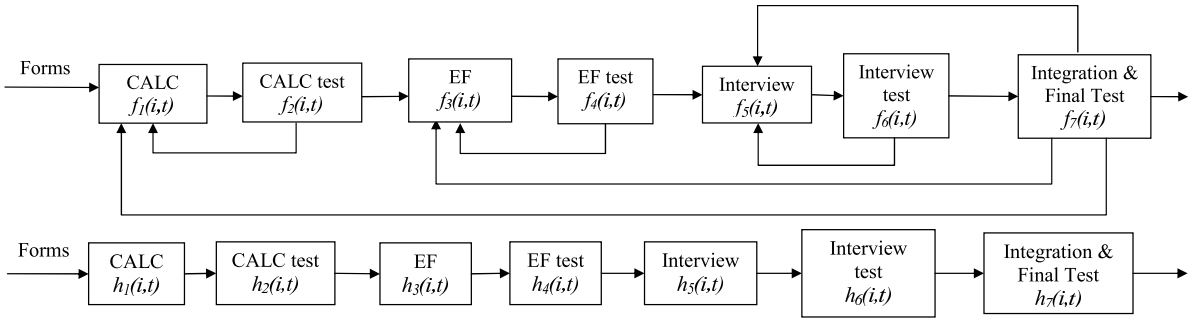

Figure 2 The original process and the no-loops approximation

unlikely. In the absence of inserted idle times, we are able to approximate the effect of feedback loops with a no-loops process, provided we suitably modify the processing times at each stage.

3.2. Approximating Feedback Loops

In the upper half of Figure 2, we show an alternate depiction of the development process that separates the sub-stages. In the lower half, we depict the no-loops approximation to the original process. Because I&FT never sends a form back to IDG, we eliminated the IDG process and its internal test from the figure.

In our approximation, the processing time distribution of a form at each stage is determined by the original processing time distribution of the form and the distribution of the number of times the form revisits the stage. In the original process, let $\rho_k(i,n)$ represent the probability that the number of times form i visits stage k is n, and let $f_k(i,t)$ denote the distribution of the processing time t of form i at stage k. In the no-loops process, the processing time distribution for form i at stage k is defined as $h_k(i,t) = \sum_n \rho_k(i,n) f_k^{(n)}(i,t)$, in which $f_k^{(n)}(i,t)$ is the n–fold convolution of $f_k(i,t)$. In Appendix B, we show that for a two-stage system with deterministic processing times, the completion time for the no-loops process converges to that of the original process as the number of forms becomes large. Our numerical experiments also show that the approximation performs well.

4. Models

Based on the foregoing assumptions, we first formulate the Monolithic Grouping and Resource Allocation Model (MGRAM) that simultaneously sorts tax forms into groups and allocates resources

to the groups. We find that MGRAM is strongly NP-hard and solvers such as Industrial Lingo are consequently ineffective in solving moderate-size problems that involve more than 500 forms. Although realistic instances of the monolithic model cannot be solved to optimality in reasonable amounts of time, discussing the formulation of the monolithic model is useful for setting the stage for the heuristic development in Section 4.2.

We then formulate three models and combine them into a hierarchical approach to heuristically solve the monolithic model. The three models we formulate calculate indices that measure the similarity between processing times for forms, use the indices to assign forms to groups, and allocate resources to the groups with the goal of releasing the software on time. This approach is similar to the classic hierarchical production planning approach in Hax and Candea (1984) if one pictures the groups of forms as families of similar items. While Hax and Candea focus on disaggregating the production plan of product types to product families and items, we are concerned with staffing the production process. The three models used in the hierarchical approach (see Figure 3) are the Grouping Index Model (GIM), the Grouping Model (GM), and the Resource Allocation Model (RAM).

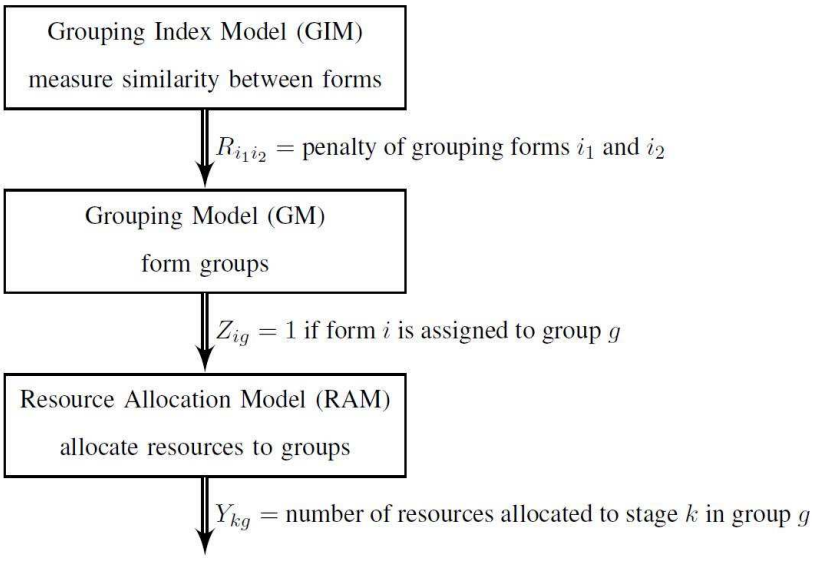

Figure 3 Models of the hierarchical approach to grouping and resource allocation

4.1. Monolithic Grouping and Resource Allocation Model (MGRAM)

Using two approaches to formulating MGRAM, we obtain an upper value (MGRAM_1) and a lower value (MGRAM_2) for the optimal number of resources. The two approaches differ in their method for approximating the time to complete the process. Let i index the set of forms $\mathcal{I}=\{1,\ldots,I\}$, k index the set of process stages $\mathcal{K}=\{1,\ldots,K\}$, and g index the set of groups $\mathcal{G}=\{1,\ldots,G\}$ to be formed. Define P_{ik} as the processing time of form i at stage k, which in fact represents the total time needed to perform a number of divisible tasks for this form. If Y_{kg} is the number of resources allocated to stage k of group g, and form i is processed in group g, then the *effective* processing time of form i at stage k is roughly $\frac{P_{ik}}{Y_{kg}}$. Let $T_{ig}=\max_{k\in\mathcal{K}}\frac{P_{ik}}{Y_{kg}}$ be the maximum effective processing time of form i across all stages in group g. A simple yet tractable approximation for the time to complete all the forms in group g is $\sum_i T_{ig}$ in which the summation encompasses all forms that are assigned to group g. We find that this estimate, motivated by Proposition 1 in the following paragraph, continues to be accurate in our numerical experiments even with feedback loops. Appendix A provides proofs for all propositions.

PROPOSITION 1. *Suppose* $G=1$ *and* $Y_k=1$ *for all* $k\in\mathcal{K}$. *Then, the time to complete the processing of forms is at most* $\sum_{i\in\mathcal{I}} max_{k\in\mathcal{K}}\{P_{ik}\}+(K-1)max_{i\in\mathcal{I}}\{max_{k\in\mathcal{K}}\{P_{ik}\}\}$ *and* $\liminf_{I\longrightarrow\infty}\frac{\sum_{i\in\mathcal{I}} max_{k\in\mathcal{K}}\{P_{ik}\}}{\sum_{i\in\mathcal{I}} max_{k\in\mathcal{K}}\{P_{ik}\}+(K-1)max_{i\in\mathcal{I}}\{max_{k\in\mathcal{K}}\{P_{ik}\}\}}=1$.

As the number of forms becomes large relative to the number of stages, $(K-1)\max_{i\in\mathcal{I}}\{\max_{k\in\mathcal{K}}\{P_{ik}\}\}$ becomes relatively small and the completion time asymptotically approaches $\sum_{i\in\mathcal{I}}\max_{k\in\mathcal{K}}\{P_{ik}\}$. In our models, each group processes at least 1,000 forms, which allows us to use $\sum_{i\in\mathcal{I}} T_{ig}$ in MGRAM_1 to approximate the time to complete all forms in group g. We set decision variable Z_{ig} equal to 1 if form i is assigned to group g, but otherwise equal to 0. Thus, we formulate MGRAM_1 as follows:

$$(\text{MGRAM}_1)\quad \min \sum_{k\in\mathcal{K}}\sum_{g\in\mathcal{G}} w_k Y_{kg} \tag{1}$$

$$s.t.$$

$$T_{ig} \geq \frac{P_{ik} Z_{ig}}{Y_{kg}} \qquad \forall i \in \mathcal{I}, \quad \forall k \in \mathcal{K}, \quad \forall g \in \mathcal{G}, \tag{2}$$

$$\sum_{i \in \mathcal{I}} T_{ig} Z_{ig} \leq D \qquad \forall g \in \mathcal{G}, \tag{3}$$

$$\sum_{g \in \mathcal{G}} Z_{ig} = 1 \qquad \forall i \in \mathcal{I}, \tag{4}$$

$$Y_{kg} \geq 1 \text{ and integer} \quad \forall k \in \mathcal{K}, \quad \forall g \in \mathcal{G}, \tag{5}$$

$$Z_{ig} \in \{0, 1\} \qquad \forall i \in \mathcal{I}, \quad \forall g \in \mathcal{G}. \tag{6}$$

Objective function (1) minimizes total resource cost. Constraint (2) defines the maximum effective processing time of form i in group g. Constraint (3) ensures that the sum of the maximum effective processing time across all stages of all forms is less than the time to deadline D in each group. Constraint (4) guarantees that each form is assigned to one group only. Finally, constraints (5) and (6) ensure that resources are positive integers and that Z_{ig} is a binary variable.

For MGRAM_2, we require that the total effective processing time for all forms *at each stage* be less than the time to deadline. In a flow shop, the maximum total effective processing time across all stages is a lower value for the completion time. The resources prescribed by MGRAM_1 will always be greater than those prescribed by MGRAM_2. The formulation of MGRAM_2 is the same as MGRAM_1, except that we replace constraints (2) and (3) with:

$$\sum_{i \in \mathcal{I}} \frac{P_{ik} Z_{ig}}{Y_{kg}} \leq D, \quad \forall k \in \mathcal{K}, \quad \forall g \in \mathcal{G}.$$

In the next proposition, we assert the complexity of the two MGRAMs.

PROPOSITION 2. *$MGRAM_1$ and $MGRAM_2$ are strongly NP-hard.*

4.2. Hierarchical Grouping and Resource Allocation Models

In this section, we formulate the grouping and resource allocation models that constitute our hierarchical approach to solving the MGRAMs heuristically. Our hierarchical models, GRAM_1 and GRAM_2, correspond to the monolithic models, MGRAM_1 and MGRAM_2. Beginning with the Grouping Index Model (GIM), we develop an index of similarity between two forms. We then use this index in the Grouping Model (GM) to assign forms to groups. Once the groups are formed, we

use the Resource Allocation Models (RAM_1 and RAM_2 corresponding to GRAM_1 and GRAM_2) to determine staffing levels.

4.2.1. Grouping Index Model (GIM)

The first component of our hierarchical framework provides a means of measuring the similarity between forms. The idea behind the GIM, which was also used in Ahmadi and Matsuo (2000), is the notion that if form i_1 and form i_2 require proportional processing times in each stage (i.e., $\frac{P_{i_1k}}{P_{i_2k}}$ is the same for all stages k) then it is suitable to assign these forms to the same group. Suppose that forms i_1 and i_2 are to be processed in one group by Y_k resources at stage k (g is suppressed) and define

$$\Lambda_{i_1} = \max_{k\in\mathcal{K}} \left\{ \frac{P_{i_1k}}{Y_k} \right\}, \quad \Lambda_{i_2} = \max_{k\in\mathcal{K}} \left\{ \frac{P_{i_2k}}{Y_k} \right\}.$$

We define *balance loss* as the total idle time arising from the difference between the maximum effective processing times and the effective processing times at other stages:

$$\sum_{k\in\mathcal{K}} \left\{ \left(\Lambda_{i_1} - P_{i_1k}/Y_k\right) + \left(\Lambda_{i_2} - P_{i_2k}/Y_k\right) \right\} Y_k = \left(\Lambda_{i_1} + \Lambda_{i_2}\right) \sum_{k\in\mathcal{K}} Y_k - \sum_{k\in\mathcal{K}} \left(P_{i_1k} + P_{i_2k}\right). \tag{7}$$

The smaller the balance loss, the more suitable it would be to place forms i_1 and i_2 in the same group. With decision variables Y_k and Λ_i, we define the GIM for forms i_1 and i_2 as

$$\text{(GIM)} \quad \min \ \left(\Lambda_{i_1} + \Lambda_{i_2}\right) \sum_{k\in\mathcal{K}} Y_k \tag{8}$$

$$s.t.$$

$$\Lambda_i \geq \frac{P_{ik}}{Y_k} \qquad i \in \{i_1, i_2\}, \quad \forall k \in \mathcal{K}, \tag{9}$$

$$Y_k \geq 1 \qquad \forall k \in \mathcal{K}.$$

The GIM addresses the following question: *Assuming we can allocate unlimited resources, what is the minimum balance loss we will incur if we assign forms i_1 and i_2 to the same group?* Note that (8) is the variable part of (7). Also, resources are allowed to take real values because the GIM is not concerned with resource allocation.

Let $\Lambda_{i_1}^*, \Lambda_{i_2}^*, Y_k^*$ be the optimal solution of (8). We define the penalty for placing forms i_1 and i_2 in the same group to be $R_{i_1 i_2} = d_{i_1 i_2} - \min_{b \neq i_1} d_{i_1 b} + d_{i_2 i_1} - \min_{b \neq i_2} d_{i_2 b}$ in which $d_{i_1 i_2} = \Lambda_{i_1}^* \sum_{k \in \mathcal{K}} Y_k^* - \sum_{k \in \mathcal{K}} P_{i_1 k}$ is the proportion of idle times attributable to form i_1, $d_{i_2 i_1} = \Lambda_{i_2}^* \sum_{k \in \mathcal{K}} Y_k^* - \sum_{k \in \mathcal{K}} P_{i_2 k}$ is the proportion of idle times attributable to form i_2, and $d_{i_1 b}$ ($d_{i_2 b}$) is the value of $d_{i_1 i_2}$ ($d_{i_2 i_1}$) when (8) is solved for form i_1 (i_2) and form $b \in \mathcal{I}$ such that $b \neq i_1$ ($b \neq i_2$).

Ideally, we would formulate and solve the GIM and minimize the balance loss for all combinations of forms. Because this would make the GIM and the GM exceedingly difficult, we find the balance loss for pairs of forms, instead, and use the sum of pairwise losses as a surrogate for the actual balance loss when more than two forms are assigned to the same group. To alleviate the issue of double counting, we subtract the terms $\min_{b \neq i_1} d_{i_1 b}$ and $\min_{b \neq i_2} d_{i_2 b}$.

4.2.2. Grouping Model (GM)

Using the indices obtained from the GIM, we formulate the GM and assign forms to groups. The lower the penalty $R_{i_1 i_2}$, the better it is to have forms i_1 and i_2 in the same group. By defining

$$X_{i_1 i_2} = \begin{cases} 1 & \text{if forms } i_1 \text{ and } i_2 \neq i_1 \text{ are assigned to the same group,} \\ 0 & \text{otherwise} \end{cases}$$

and the total processing time of form i as $P_i = \sum_{k \in \mathcal{K}} P_{ik}$, we formulate the GM with decision variables $X_{i_1 i_2}$ and Z_{ig} as

$$\text{(GM)} \min \sum_{i_1=1}^{I-1} \sum_{i_2=i_1+1}^{I} R_{i_1 i_2} X_{i_1 i_2} \tag{10}$$

$$s.t.$$

$$X_{i_1 i_2} \geq Z_{i_1 g} + Z_{i_2 g} - 1 \qquad \forall g \in \mathcal{G},\ \forall i_1, i_2 \in \mathcal{I},\ i_1 \neq i_2, \tag{11}$$

$$\sum_{g \in \mathcal{G}} Z_{ig} = 1 \qquad \forall i \in \mathcal{I}, \tag{12}$$

$$\sum_{i \in \mathcal{I}} P_i Z_{ig} \leq Q \qquad \forall g \in \mathcal{G}, \tag{13}$$

$$X_{i_1 i_2} \in \{0,1\} \qquad \forall i_1, i_2 \in \mathcal{I},\ i_1 \neq i_2, \tag{14}$$

$$Z_{ig} \in \{0,1\} \qquad \forall i \in \mathcal{I},\ \forall g \in \mathcal{G}, \tag{15}$$

in which $Q = (1+\delta) \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} P_{ik} / G$.

Constraint (13) requires that the total processing time of forms in each group not exceed the average total processing time per group by more than a predefined fraction δ, which is a parameter that controls the feasibility of the GM. If δ is too low, the GM may be infeasible. If δ is too large, then (13) becomes a redundant constraint. Although the value of δ need not be unique, our numerical investigations indicate that 0.25 is reasonable for three to five groups (see Appendix C.1). However, when the number of groups increases, δ should be increased to ensure feasibility. We can now attest to the complexity of the GM with Proposition 3.

PROPOSITION 3. *The GM is strongly NP-complete.*

Given that the GM is a hard problem, we use a decomposition procedure described in Section 5 to heuristically solve it.

4.2.3. Resource Allocation Models (RAM)

For the last component of the hierarchical framework, we formulate RAM_1 and RAM_2 to find upper and lower values for staffing levels that allow each group to finish processing their assigned forms on time at minimum expense. The formulation of RAM_1 and RAM_2 is the same for all groups. Therefore, we suppress g in decision variables Y_{kg} and T_{ig}, and, with a little abuse of notation, use $\mathcal{I}$ in this section for the set of forms that is assigned to the same group. RAM_1 is formulated as

$$
\begin{aligned}
(\text{RAM}_1)\ \min\ & \sum_{k\in\mathcal{K}} w_k Y_k \\
s.t.\ & \\
& T_i \geq \frac{P_{ik}}{Y_k} && \forall k\in\mathcal{K},\ \ \forall i\in\mathcal{I}, && (16)\\
& \sum_{i\in\mathcal{I}} T_i \leq D, && && (17)\\
& Y_k \geq 1 \text{ and integer}, && \forall k\in\mathcal{K}. &&
\end{aligned}
$$

We obtain RAM_2 from RAM_1 by replacing (16) and (17) with $\sum_{i\in\mathcal{I}} P_{ik}/Y_k \leq D$ for all stages k. The following proposition states that RAM_1 is a hard problem.

PROPOSITION 4. *RAM_1 is binary NP-hard.*

In Section 5, we propose a pseudo-polynomial time algorithm to solve RAM_1, and show that, unlike RAM_1, RAM_2 is an easy problem.

4.3. Process Line Separation Model (PLSM)

The hierarchical models we have described thus far assume that TSDC will acquire additional resources if necessary. Since hiring new employees requires time to interview candidates and train new hires, TSDC managers were also concerned with managing their existing resources. More specifically, they were interested in a model that would distribute the existing resources to two major process lines: one line for processing all federal forms and one line for processing all state forms. The Process Line Separation Model (PLSM) addresses TSDC's concern.

Let M_k be the number of existing resources at stage k. Also, let $\mathcal{I}_\mathcal{S}$ and $\mathcal{I}_\mathcal{F}$ denote the set of federal and state forms indexed by i_s and i_f. We use Y_{k_s} and Y_{k_f} to denote the number of resources allocated to stage k to process state and federal forms. With decision variables Y_{k_s}, Y_{k_f}, T_{i_s}, and T_{i_f}, the PLSM is formulated as

$$\text{(PLSM)} \quad \min \max \left\{ \sum_{i_s \in \mathcal{I}_\mathcal{S}} T_{i_s}, \sum_{i_f \in \mathcal{I}_\mathcal{F}} T_{i_f} \right\} \tag{18}$$

s.t.

$$T_{i_s} \geq \frac{P_{i_s k}}{Y_{k_s}} \qquad \forall k \in \mathcal{K}, \ \ \forall i_s \in \mathcal{I}_\mathcal{S}, \tag{19}$$

$$T_{i_f} \geq \frac{P_{i_f k}}{Y_{k_f}} \qquad \forall k \in \mathcal{K}, \ \ \forall i_f \in \mathcal{I}_\mathcal{F}, \tag{20}$$

$$Y_{k_s} + Y_{k_f} \leq M_k \qquad \forall k \in \mathcal{K}, \tag{21}$$

$$Y_{k_s}, Y_{k_f} \geq 1 \text{ and integer} \qquad \forall k \in \mathcal{K}.$$

Objective function (18) minimizes the approximate process completion time. Constraints (19) and (20) define the maximum effective processing time of state and federal forms. Constraint (21) represents resource availability.

Proposition 5. *PLSM is binary NP-hard.*

5. Solution Procedures

In this section, we describe the solution procedures for the hierarchical models and the PLSM.

5.1. GIM Solution

The GIM can be solved by defining $\beta = \frac{\Lambda_{i_1}}{\Lambda_{i_1}+\Lambda_{i_2}}$ and finding Y_k from (9) as follows:

$$\begin{aligned}\min\ (\Lambda_{i_1}+\Lambda_{i_2})\sum_{k\in\mathcal{K}} Y_k &\equiv \min_{\Lambda_{i_1},\Lambda_{i_2}>0}(\Lambda_{i_1}+\Lambda_{i_2})\sum_{k\in\mathcal{K}}\max\left(P_{i_1k}/\Lambda_{i_1}, P_{i_2k}/\Lambda_{i_2}\right)\\ &\equiv \min_{0<\beta<1}\sum_{k\in\mathcal{K}}\max\ \left(P_{i_1k}/\beta, P_{i_2k}/(1-\beta)\right).\end{aligned}$$

The function $\max\left(P_{i_1k}/\beta, P_{i_2k}/(1-\beta)\right)$ is strictly convex in $\beta \in (0,1)$. Because the objective function is the sum of strictly convex functions, we easily find the optimal solution.

5.2. GM Solution

To solve the GM, we use a decomposition procedure that provides a lower bound on the objective function and offers feasible solutions. If we relax (11) by positive Lagrangian multipliers $\gamma_{i_1i_2g}$, we get

$$\begin{aligned}L(\gamma) = \min\ &\sum_{i_1=1}^{I-1}\sum_{i_2=i_1+1}^{I}\left(R_{i_1i_2}-\sum_{g\in\mathcal{G}}\gamma_{i_1i_2g}\right)X_{i_1i_2}+\sum_{g\in\mathcal{G}}\sum_{i_1=1}^{I}\left(\sum_{i_2=1}^{i_1-1}\gamma_{i_2i_1g}+\sum_{i_2=i_1+1}^{I}\gamma_{i_1i_2g}\right)Z_{i_1g}\\ &-\sum_{g\in\mathcal{G}}\sum_{i_1=1}^{I-1}\sum_{i_2=i_1+1}^{I}\gamma_{i_1i_2g}\\ s.t.&\\ &(12),(13),(14),(15).\end{aligned}$$

For a vector of Lagrangian multipliers, γ, with $\gamma \geq 0$, the math program that defines the function $L(\gamma)$ can be decomposed into two math programs:

$$\begin{aligned}L_1(\gamma) = \min\ &\sum_{i_1=1}^{I-1}\sum_{i_2=i_1+1}^{I}\left(R_{i_1i_2}-\sum_{g\in\mathcal{G}}\gamma_{i_1i_2g}\right)X_{i_1i_2}\\ s.t.&\\ &(14),\end{aligned}\qquad\qquad\begin{aligned}L_2(\gamma) = \min\ &\sum_{g\in\mathcal{G}}\sum_{i_1=1}^{I}\theta_{i_1g}Z_{i_1g}\\ s.t.&\\ &(12),(13),(15),\end{aligned}$$

in which $\theta_{i_1 g} = \sum_{i_2=1}^{i_1-1} \gamma_{i_2 i_1 g} + \sum_{i_2=i_1+1}^{I} \gamma_{i_1 i_2 g}$.

In the math program that defines the function $L_1(\gamma)$, the optimal value of each $X_{i_1 i_2}$ is equal to 0 if its coefficient in the objective function is positive; otherwise it is equal to 1. The math program that defines the function $L_2(\gamma)$ is a packing-by-cost variation of the bin packing problem in which a set of items should be packed in bins (groups) of the same capacity to minimize the total packing cost. We solve the bin packing subproblem using a dynamic program in which $V_i(U_1, U_2, ..., U_G)$ denotes the minimum cost of assigning form i to one of the groups with sufficient capacity, given that the remaining capacity of bin (group) g is U_g and forms 1 to $i-1$ are already assigned. The value functions are calculated as follows:

$$V_i(U_1, U_2, ..., U_G) = \min_{\{g \in \mathcal{G} | U_g \geq P_i\}} \{\theta_{ig} + V_{i+1}(U_1, U_2, ..., U_g - P_i, ..., U_G)\} \quad \forall i \in \mathcal{I}.$$

To ensure feasibility, we set $V_i(U_1, U_2, ..., U_G) = +\infty$, if $U_g < P_i$ for all $g \in \mathcal{G}$. The optimal solution to $L_2(\gamma)$ is $V_1(Q, Q, ..., Q)$ and the complexity of the procedure is $O(IQ^G)$. This complexity is practically manageable because the number of groups does not exceed five.

Since $L(\gamma)$ is a lower bound on (10), we solve the following math program, using a subgradient optimization algorithm (Fisher 1981, 1985) to find the best of such lower bounds:

$$\begin{aligned} \max \quad & L(\gamma) \\ s.t. \quad & \\ & \gamma_{i_1 i_2 g} \geq 0, \quad \forall i_1, i_2 \in \mathcal{I}, \forall g \in \mathcal{G}. \end{aligned}$$

In addition to obtaining a lower bound on the objective function, we also use the decomposition procedure to find feasible solutions to the GM. In each iteration of the subgradient optimization algorithm, we can construct a feasible solution to the GM by using the optimal values of $Z_{i_1 g}$ and $Z_{i_2 g}$ and setting $X_{i_1 i_2} = \max_{g \in \mathcal{G}} \{Z_{i_1 g} + Z_{i_2 g} - 1\}$. These feasible grouping scenarios become inputs to the RAMs for obtaining the optimal resource allocation.

5.3. RAM_1 and RAM_2 Solutions

We first explain how to solve RAM_2 because it is an easy problem. The optimal number of resources at stage k in RAM_2 is $Y_k^* = \left\lceil \sum_{i \in \mathcal{I}} P_{ik} / D \right\rceil$, for which $\lceil x \rceil$ is the smallest integer larger than or equal to x.

RAM_1 can be transformed to a shortest path problem. First, we find a lower value Y_k^{min} and an upper value Y_k^{max} for the optimal solution, $Y_k^*, k \in \mathcal{K}$, to limit the size of the network. The lower value for Y_k^* can be found by replacing (16) with $\sum_{i\in\mathcal{I}} T_i \geq \sum_{i\in\mathcal{I}} P_{ik}/Y_k$ and enforcing (17). Thus, $Y_k^* \geq Y_k^{min} = \left\lceil \sum_{i\in\mathcal{I}} P_{ik}/D \right\rceil$ for all stages $k \in \mathcal{K}$. For the upper value, set $P_{ik} = \max_{k\in\mathcal{K}}\{P_{ik}\}$ for all stages $k \in \mathcal{K}$ to inflate the processing times of form $i \in \mathcal{I}$ to its maximum processing time across all stages. Then, if $Y_k = \overline{Y} = \sum_{i\in\mathcal{I}} \max_{k\in\mathcal{K}}\{P_{ik}\}/D$ for all stages $k \in \mathcal{K}$, the deadline will be met. Therefore, $\sum_{k\in\mathcal{K}} w_k \overline{Y} \geq \sum_{k\in\mathcal{K}} w_k Y_k^*$, which means $Y_k^* \leq Y_k^{max} = \left\lceil \left(\overline{Y} \sum_{k\in\mathcal{K}} w_k - \sum_{r\in\mathcal{K}, r\neq k} w_r Y_r^{min} \right) / w_k \right\rceil$ for all stages $k \in \mathcal{K}$.

The network for RAM_1 has K layers and each node is represented by (k, Y_k, E) for which $E = \sum_{i\in\mathcal{I}} T_i - D$ given resources $Y_1, \ldots, Y_K$. The network is constructed using the following steps.

Step 0. Generate start (0) and finish (F) nodes.

Step 1. Generate nodes $(1, Y_1, E)$ starting from $Y_1 = Y_1^{min}$ and increasing by 1 while setting $Y_r = Y_r^{min}$ for stages $r > 1$. The cost of arc $(0) \to (1, Y_1, E)$ is $(Y_1 - Y_1^{min})\omega_1$. Stop adding new nodes if $Y_1 = Y_1^{max}$ or $E \leq 0$. If $E \leq 0$ occurs first, connect the last node to node F with cost 0.

Step 2. In layers $k = 2, \ldots, K$, generate the children of nodes in layer $k-1$ with $E > 0$ in the same manner as Step 1. To be more specific, the value of E in node (k, Y_k, E) is computed by setting $Y_r = Y_r^{min}$ for stages $r > k$ and setting Y_r for stages $r \leq k$ equal to their values on the path $(1, Y_1, E) \to (2, Y_2, E) \to \cdots \to (k-1, Y_{k-1}, E) \to (k, Y_k, E)$. The cost of arc $(k-1, Y_{k-1}, E) \to (k, Y_k, E)$ is $(Y_k - Y_k^{min})\omega_k$. In layer K, the cost of $(K, Y_K, E) \to F$ will be a very large number if $E > 0$ and 0 otherwise.

Step 3. Compute the shortest path of the network. The optimal solution to RAM_1 is the sum of the cost of the shortest path and the cost of allocating $Y_k^{\min}$ to each stage. The optimal resource configuration can be determined by traversing the shortest path.

The complexity of the size of the network is $O\left(\left(\max_{k\in\mathcal{K}}\{Y_k^{max} - Y_k^{min}\}\right)^K\right)$. Appendix D provides an example of the network construction.

5.4. PLSM Solution

To solve the PLSM, we propose a heuristic solution that relaxes the integrality constraint on Y_{kg}, after which we establish the worst-case performance of the heuristic. Let $\widetilde{Y}_{k_s}$ and $\widetilde{Y}_{k_f}$ be the solution to the PLSM when the integrality constraints are relaxed. We find an integer solution using the following algorithm.

Step 1. For all stages $k \in \mathcal{K}$, set $Y_{k_s} = \lfloor \widetilde{Y}_{k_s} \rfloor$ and $Y_{k_f} = \lfloor \widetilde{Y}_{k_f} \rfloor$, for which $\lfloor x \rfloor$ is the greatest integer less than or equal to x, and define $\overline{M}_k = \max\left(0, M_k - Y_{k_s} - Y_{k_f}\right)$ as the number of remaining available resources in stage $k \in \mathcal{K}$.

Step 2. Find the first stage k with $\overline{M}_k > 0$. First compute TS_k, the objective function of the PLSM for $\left(Y_{k_s}+1, Y_{k_f}\right)$. Then compute TF_k, the objective function of the PLSM for $\left(Y_{k_s}, Y_{k_f}+1\right)$. If $TS_k < TF_k$, let $Y_{k_s} := Y_{k_s}+1$; otherwise, let $Y_{k_f} := Y_{k_f}+1$. Update $\overline{M}_k$ and repeat this step until $\overline{M}_k = 0$. Find the next stage with $\overline{M}_k > 0$ and repeat this step until $\overline{M}_k = 0$ for all $k \in \mathcal{K}$.

Let ϕ^H be the objective function value of the above heuristic. Proposition 6 places a bound on the worst-case performance of the heuristic.

PROPOSITION 6. *If ϕ^* denotes the optimal value of the PLSM, then* $\dfrac{\phi^H}{\phi^*} \leq 2$.

6. Numerical Experiments on Performance

In this section, we report the results of our numerical experiments for evaluating the proposed hierarchical procedure. We chose the parameters for the experiments based our observations at TSDC. We found that historically about 50% of the forms are easy to process, 20% are difficult to process, and the remaining 30% are moderately difficult to process. Table 1 shows the processing time intervals (in hours) for these three categories, with the last row displaying the scaled resource costs in thousands of dollars.

For our experiments, we calculated the number of working hours in the time interval between the start of the project on August 1^{st} and the finish on December 15^{th}. In every experiment, we considered different combinations of group G and stage K, and we generated 90 instances of the models for each combination. We evaluated the quality of the proposed hierarchical approach

Table 1 Processing time intervals for form categories and resource costs at each stage

	Processing Time Intervals (hours)									
Form Categories	IDG Process	IDG Test	CALC Process	CALC Test	EF Process	EF Test	Interview Process	Interview Test	Integration	Final Test
Easy	[2 5]	[1 2]	[3 6]	[1 2]	[2 5]	[0.5 1]	[1 3]	[0.5 1]	[0.5 1.5]	[1 2]
Moderately difficult	[5 12]	[1.5 2.5]	[8 12]	[2 3]	[4 7]	[0.5 1.5]	[2 5]	[0.5 1.5]	[1 1.5]	[2 3]
Difficult	[12 20]	[2 3]	[12 24]	[3 4]	[6 9]	[1 1.5]	[4 7]	[1 1.5]	[1.5 2]	[3 4]
Cost ($1000)	50	10	60	15	40	10	35	10	50	15

by means of two major comparisons. First, we compared the solution of the hierarchical models to the solution of the monolithic models, for small to moderate instances that we could solve optimally. Second, we compared the solution of the hierarchical models to the solution obtained from simulation–optimization.

6.1. Hierarchical vs. Monolithic

Table 2 displays the average ratio of the total workforce cost of the hierarchical models to that of the monolithic models, Ave(HA/MO), calculated from 810 problem instances. The bottom three rows of the table provide the grand averages and 95% confidence intervals (CI) for Ave(HA/MO). The grand averages indicate that the total workforce cost of the hierarchical models are on average 2.07% greater than the total workforce cost of the monolithic models. We also conducted an experiment to compare the performance of the hierarchical models to that of the monolithic models when the problem size grows. Figure 4 shows Ave(HA/MO) for 10 instances of the hierarchical models when $G = 5$, $K = 10$, and I increases to 500. The results indicate that the quality of the hierarchical solution does not deteriorate when we increase the size of the models. Therefore, the hierarchical models provide a good heuristic solution to the complex monolithic models.

6.2. Hierarchical vs. Simulation-Optimization

Next, we compared the result of our hierarchical approach to the simulation–optimization approach in a fashion similar to that used in Kekre et al. (2009). We used Arena simulation software to

Table 2 The hierarchical models vs. the monolithic models

G	K	Ave(HA/MO)					
		$GRAM_1/MGRAM_1$			$GRAM_2/MGRAM_2$		
		$I=50$	$I=75$	$I=100$	$I=50$	$I=75$	$I=100$
3	8	1.022	1.014	1.023	1.011	1.024	1.035
	9	1.011	1.021	1.032	1.016	1.031	1.046
	10	1.024	1.009	1.019	1.034	1.030	1.016
4	8	1.007	1.027	1.024	1.013	1.020	1.005
	9	1.005	1.010	1.012	1.024	1.024	1.040
	10	1.021	1.026	1.024	1.013	1.013	1.015
5	8	1.024	1.028	1.038	1.012	1.022	1.025
	9	1.021	1.027	1.027	1.009	1.008	1.025
	10	1.028	1.016	1.016	1.015	1.021	1.008
CI (lower)		1.017	1.019	1.022	1.015	1.020	1.022
Average		1.018	1.020	1.024	1.016	1.022	1.024
CI (upper)		1.019	1.021	1.025	1.017	1.023	1.025

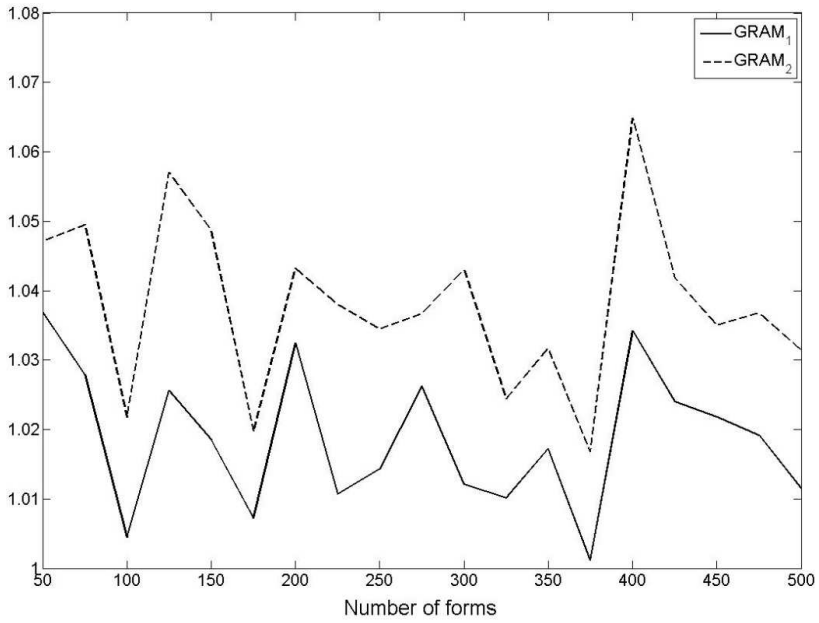

Figure 4 Ratio of the solution of the hierarchical models to the solution of the monolithic models

capture the operational particulars of the system and to compute the exact completion time of the process. The arrival times of the forms, processing times, and probability of feedback are all random numbers. Because our hierarchical models approximate the completion time, arrival times, and feedback loops, the simulation model scheduled overtime when was needed to meet the deadline. The overtime cost was assumed to be 50% higher than the cost of regular time.

For simulation–optimization, we used Arena's OptQuest in an iterative manner. OptQuest gen-

erated a vector of resource configuration and assignment of forms to groups, and Arena evaluated the project completion time and the total workforce cost. We repeated this process and used the default options of OptQuest to stop simulation–optimization.

Table 3 reports the average ratio of the total workforce cost of the hierarchical models to the total workforce cost obtained from simulation–optimization for small instances, Ave (HA/SO). We should note that the ratios for $GRAM_2$ are higher than the ratios for $GRAM_1$, because $GRAM_2$ allocates fewer resources to each stage than $GRAM_1$ and consequently results in more overtime. Although simulation–optimization achieves a lower overall cost than $GRAM_1$, the total workforce cost of $GRAM_1$ is on average only 2.57% greater than the total workforce cost obtained from simulation–optimization.

Table 3 The hierarchical models vs. simulation–optimization

G	K	Ave(HA/SO)					
		$GRAM_1$/SO			$GRAM_2$/SO		
		$I=50$	$I=75$	$I=100$	$I=50$	$I=75$	$I=100$
3	8	1.021	1.021	1.037	1.103	1.064	1.105
	9	1.029	1.023	1.033	1.066	1.065	1.062
	10	1.009	1.016	1.026	1.027	1.022	1.058
4	8	1.042	1.010	1.033	1.053	1.092	1.065
	9	1.032	1.033	1.029	1.062	1.086	1.065
	10	1.043	1.022	1.024	1.042	1.033	1.063
5	8	1.024	1.025	1.025	1.042	1.102	1.049
	9	1.016	1.033	1.018	1.045	1.030	1.034
	10	1.014	1.036	1.021	1.076	1.041	1.107
CI (lower)		1.024	1.022	1.025	1.053	1.055	1.063
Average		1.026	1.024	1.027	1.057	1.059	1.068
CI (upper)		1.027	1.025	1.028	1.060	1.062	1.070

Although simulation–optimization appears to provide better solutions for smaller instances, it is computationally very intensive. We were unable to use simulation–optimization due to the large number of binary variables that we encountered in our problem. Figure 5 shows the ratio of simulation–optimization runtime to the runtime of the hierarchical models for different number of forms, $G=3$, and $K=8$. The ratio of the runtime grows almost exponentially with the number of forms. Furthermore, the number of decision variables exceeded the current capabilities of commercial software such as OptQuest and prohibited us from using this methodology in practice. We

have also evaluated the computation time of the hierarchical models. Our experiments indicate that our models have a satisfactory runtime (see Appendix C.2).

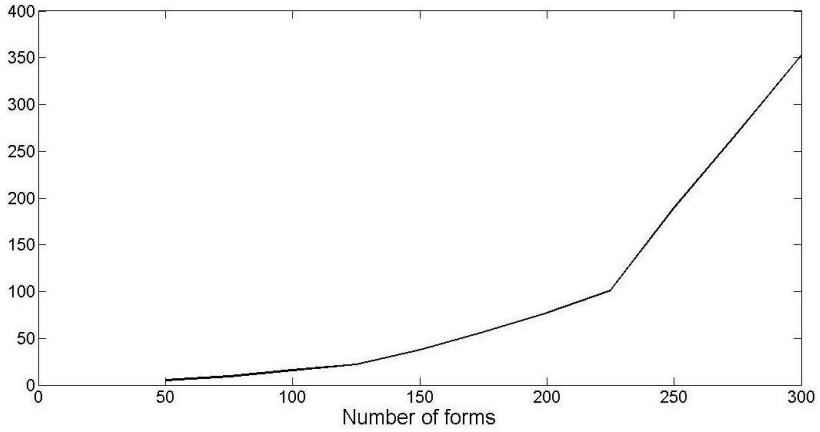

Figure 5 Ratio of simulation–optimization runtime to hierarchical approach runtime

6.3. Effect of Approximations

In addition to the aforementioned experiment, we also used simulation–optimization to examine the effect of approximations used by the hierarchical procedure on total workforce cost. In particular, we were interested to know what percentage of the error in the total workforce cost of GRAM_1 was due to approximating feedback loops and what percentage was due to approximating the process completion time with $\max_{g\in\mathcal{G}}\{\sum_{i\in\mathcal{I}} T_{ig}Z_{ig}\}$. For this purpose, we conducted two variations of simulation–optimization for the GRAM_1 instances in Table 3 and calculated the ratio, λ, for each variation. The denominator of λ in both variations was the average difference between the total workforce cost of GRAM_1 and the total workforce cost of the simulation–optimization experiment in Section 6.2.

In the first variation of our simulation–optimization, we eliminated the feedback loops and applied the approximation described in Section 3.2 to the process. We then used the difference between the total workforce cost of the new simulation–optimization experiment and the total workforce cost of GRAM_1 as the numerator of λ, and found that $37.6\% \pm 2.3\%$ (95% confidence

interval) of the average difference between the total workforce cost of $GRAM_1$ and the simulation–optimization in Section 6.2 was due to the feedback loop approximation.

In the second variation, we applied the approximation for feedback loops and approximated dynamic arrivals with the availability of all forms when the process started. We used the difference between the total workforce cost of this simulation–optimization experiment and the total workforce cost of $GRAM_1$ as the numerator of λ. The resulting ratio indicated that $52.3\% \pm 2.9\%$ (95% confidence interval) of the average difference between the total workforce costs in Section 6.2 was due to approximating the process completion time in $GRAM_1$.

In summary, we conclude that $GRAM_1$ generates very good solutions for organizing the complex development process, and that the solutions of $GRAM_1$ and $GRAM_2$ provide TSDC managers with a range of resource configurations to finish the processing of forms in all groups by the deadline.

7. Implementation

Prior efforts at process improvement had rendered TSDC receptive to our analytical approach. Upon examining our analysis and considering the potential benefits to TSDC, the Vice President of Operations directed the groups to implement our models during the 2010 development period. Process managers cooperated fully and provided us with the necessary data, and the vice president removed internal obstacles throughout the project, making her trust and commitment essential to testing and implement our solutions in a real-world situation.

7.1. Simulations

Prior to putting our hierarchical models into practice at TSDC, we evaluated the quality of the solutions they evoked by collecting historical data and then developing a simulation model. To estimate the expected number of forms, we fit arrival data from the past several decades to a linear regression model. To generate arrival patterns, we used the average cumulative arrivals of forms over the past three years (2007-2009). Then, we randomly assigned arrival dates to them such that their cumulative arrivals matched the historical percentage of arrivals. To estimate processing time

distributions at each stage, we designed a questionnaire (see Appendix E) asking employees and managers to provide estimates of the following:

- Minimum, average, and maximum processing times needed to complete the forms at each stage;
- Percentage of forms that take less than 25%, between 25% and 50%, between 50% and 75%, and above 75% of the maximum processing time; and
- Percentage of forms that fail each internal test and Final Test and return for rework.

Subsequently, we developed a simulation model that accounted for uncertainty in the arrival and processing times of forms, feedback loops for addressing errors (bug fixes), and TSDC rules for sequencing forms and scheduling overtime policies. At the time of our study, TSDC employed two sequencing rules: (1) among available forms, the form with the shortest total processing time, $\min_{i\in\mathcal{I}}\left\{\sum_{k\in\mathcal{K}} P_{ik}\right\}$, was assigned to a resource first; and (2) at each stage, forms were assigned to the resource with the least amount of work to do. TSDC overtime policy required managers to update their estimates of total remaining work at the end of each week. They determined the amount of overtime for the upcoming weeks by calculating the difference between new and old estimates and dividing it by the number of weeks remaining until the deadline.

To validate the simulation model, we applied a procedure similar to that used in Kekre et al. (2009). We ran the simulation model using historical data on staffing levels for a two-year period, and the model provided values for the total amount of work remaining at the end of each week. Next, we compared these weekly values with the actual total amount of remaining work at the end of each week provided by management. We constructed a 95% paired$-t$ confidence interval $\overline{\chi} \pm t_{(N-1,0.975)}\sqrt{S_\chi^2/N}$ in which $\overline{\chi}$ and S_χ denote the average and sample standard deviation of the difference between actual and simulated amount of work remaining, and $t_{(N-1,0.975)}$ denotes the 0.975 critical value for the t distribution with $N-1=42$ degrees of freedom. Because the confidence interval was $[-0.012, 0.021]$ and included 0, we could not reject the null hypothesis that the average of the total work to do at the end of each week obtained from the simulation model

was the same as the actual average. Thus we concluded that our simulation model was a good representation of the actual process.

We then conducted the runs test (Black 2011) at the 5% significance level to discover whether or not errors in the simulated total amount of work to do at the end of each week were random. We arrived at a test statistic value of -1.31, which falls between -1.96 and 1.96. Therefore, we could not reject the null hypothesis that the errors were random.

7.2. Two-Phase Implementation

TSDC managers requested that we implement the models in two phases. The goal of phase 1 was to assess the potential benefits of adopting the models' recommendations without altering the workforce level. The goal of phase 2 was to implement these recommendations by hiring and relocating employees during the 2010 production season.

Phase 1 entailed analyzing the benefits of optimally dividing the existing workforce into two designated groups: one for all state forms and one for all federal forms. We were given 2009 TSDC data for the number of resources at each stage, M_k, and the processing times of each form, P_{ik}. Because P_{ik} already included all processing and reprocessing times, we did not apply our approximation procedure for feedback loops in this phase. We used the PLSM to allocate resources to these two predefined groups. Next, we incorporated the actual arrival pattern of forms in 2009 and the solution from the PLSM into the simulation model and evaluated the total time needed to process the forms in each of the two groups. We compared the simulated process completion time to the actual completion time in 2009 and found that TSDC could have reduced the completion time by 23.5%.

Phase 2 involved fully implementing our models throughout the 2010 tax season. As mentioned earlier, a regression model generated the number of forms and questionnaires collected processing time estimates. We followed the procedure in Section 3.2 to calculate the no-loops approximate processing times at each stage. From discussions about the relative importance of resource pooling, communication, and coordinating and controlling the workflow, the process managers elected to

use two groups for processing all forms. We used $GRAM_1$ and $GRAM_2$ to generate a minimum and maximum staffing level for each group at each stage. TSDC proceeded to base hiring and relocation decisions for each stage on the average staffing level obtained from $GRAM_1$ and $GRAM_2$.

In accordance with our suggested new configuration, TSDC relocated approximately 12% of the 237 employees to new job assignments and hired 8 new employees (a 3% increase in the workforce). Classified into three major skill categories—programmers, testers, and accountants—employees can be relocated within their category but not across categories. Therefore, of the 12% who were relocated, 8.5% were relocated from other teams working on the same product and 3.5% came from a different product line.

7.3. Outcomes

To measure the savings obtained by implementing the hierarchical models, TSDC compared overtime and total cost figures for the 2010 tax year to those from 2009. The company discovered that it enjoyed a 25.7% reduction in overtime and an 11.3% reduction in total workforce cost. The high cost of engaging eight new employees notwithstanding, the hiring and relocation decisions helped reduce overtime payments, which ultimately reduced total workforce cost. The savings proved even greater when we considered the total amount of work before and after release. In 2009, total overtime was 59,439 hours, and the ratio of total overtime to regular hours was 22.5%. In 2010, total overtime declined by 31.6% to 40,656 hours, and the ratio to regular hours was 15.4%. The total workforce cost in 2010 was 13.6% lower than that of 2009, roughly $960,000 in cost savings. Precise workforce costs being confidential, TSDC did confirm that the savings were not the product of other factors such as employee turnover or changing product demand, workforce skills, or structure of the software. In fact, the actual amount of work TSDC accomplished in 2010 was 1.8% greater than that in 2009 due to changes in the forms. Therefore, we can claim that the savings were the result of using our decision-making tools.

At the end of the season, we reapplied our hierarchical models and used the simulation model to understand the contribution different decisions made to the savings. We found that with two groups

and hiring not allowed, the savings in total overtime and workforce cost would have been reduced to 22.8% and $573,000. We also found that with one group processing all forms and hiring allowed, the savings would have reduced to 21.9% and $687,000. It is important to note that there are a lot of intangible and non-monetary benefits in organizing the development process by creating groups. Finally, if relocating employees was not allowed, the savings would have been 18.5% and $660,000. Motivated by the savings in overtime and total resource costs, TSDC decided to implement our models every year.

In addition to saving money and completing the software on time, our models helped TSDC resolve some long-standing internal disagreements about task assignment and workforce management. Some managers were particularly amenable to our proposed solutions because they did not involve issuing pink slips. Also, establishing two processing lines promoted healthy competition between groups to complete tasks sooner with less overtime.

One challenge to implementation was estimating processing times. At the end of the season, we found that the actual total amount of work was 3.6% larger than our estimates. This error was due to estimating the number and processing times of forms and approximating the effect of feedback loops. When we reapplied our models at the end of the 2010 season using the actual processing times, we found that the total workforce cost savings could have been $1,124,000. In an effort to improve accuracy, TSDC decided to improve the system for recording processing times and rework iterations.

TSDC managers are now contemplating expanding our modeling framework to development processes for other product lines. Because employees with certain skills (e.g., programmers) can work in across products, a consolidated workforce management system could help TSDC utilize its personnel more efficiently.

8. Concluding Remarks

The survival of a commercial tax preparation application depends on developers being able to accurately update thousands of individual forms under extremely tight release deadlines. Studying

the software development process at one large company, we observed that dynamically-arriving form changes, variable processing times, feedback loops, and high task volume make process management a formidable undertaking. The same challenges are faced by many companies servicing highly-regulated domains that require them to routinely upgrade complex software applications.

We used an approximation to capture the effect of feedback loops on completion time. To help the company manage the development process more effectively, we then introduced a hierarchical framework that focused on assigning tax forms to groups and allocating resources to meet the release deadline. Numerical experiments supported our modeling assumptions and attested to the excellent quality of the hierarchical models and solution procedures. Implementing our models reduced overtime hours by 31% and total workforce cost by 13% or around $1 million. The software company successfully delivered the application on time, even though the amount of work performed was greater than in the previous year. We hope to study other product lines at the company and expand our models to manage more processes.

Acknowledgments

The authors sincerely thank the area editor, the associate editor, and the two anonymous reviewers for their helpful and constructive comments and suggestions. The financial support from the UCLA Harold and Pauline Price Center for Entrepreneurial Studies is greatly appreciated.

References

Adler, P., A. Mandelbaum, V. Nguyen, E. Schwerer. 1992. From project to process management in engineering: strategies for improving development cycle time. *Sloan School of Management,* MIT.

Ahmadi, R., H. Matsuo. 2000. A mini-line approach to pull production. *Eur. J. Oper. Res.* **125** 340–358.

Ahmadi, R., T. A. Roemer, R. H. Wang. 2001. Structuring product development processes. *Eur. J. Oper. Res.* **130** 539–558.

Asundi, J., S. Sarkar. 2005. Staffing software maintenance and support projects. *Proceedings of 38th Hawaii International Conference on System Sciences.* Hawaii, 1–8.

Bureau of Labor Statistics. 2012. Industry employment and output projections to 2020. Accessed on July 14, 2012 at `http://www.bls.gov/opub/mlr/2012/01/art4full.pdf`

Black, K. 2011. *Business Statistics: For Contemporary Decision Making.* Wiley, Hoboken, NJ.

Brooks, F. P. 1975. *The Mythical Man-Month: Essays on Software Engineering.* Addison–Wesley, Reading, MA.

Browning, T. R., R. V. Ramasesh. 2007. A survey of activity network-based process models for managing product development projects. *Prod. Oper. Management.* **16**(2) 217–240.

Carrascosa, M., S. D. Eppinger, D. E. Whitney. 1998. Using the design structure matrix to estimate product development time. *ASME Design Automation Conference.* Atlanta, GA.

Cusumano, M. A. 1997. How MicroSoft makes large teams work like small teams. *Sloan Management Review.* **39**(1) 9–20.

Cusumano, M., A. MacCormack, C. F. Kemerer, B. Crandall. 2003. Software development worldwide: The state of the practice. *IEEE Software,* **20** 28–34.

Dawande, M., M. Johar, S. Kumar, V. S. Mookerjee. 2008. A comparison of pair versus solo programming under different objectives: An analytical approach. *Inf. Syst. Res.* **19**(1) 71–92.

Department of Justice. 2011. Assistant Attorney General Conference Call. Accessed on September 26, 2011 at `http://www.justice.gov/iso/opa/atr/speeches/2011/at-speech-110523.html`

Electronic Tax Administration Advisory Committee. 2011. Annual Report to the Congeress. Accessed on September 26, 2011 at `http://www.irs.gov/pub/irs-pdf/p3415.pdf`

Feng, Q., V. S. Mookerjee, S. P. Sethi. 2006. Optimal policies for the sizing and timing of software maintenance projects. *Eur. J. Oper. Res.* **173** 1047–1066.

Graves S. C., H. C. Meal, D. Stefek, A. H. Zeghmi. 1983. Scheduling of re-entrant flow shops . *J. of Oper. Management.* **3**(4) 197–207.

Hax A. C., D Candea. 1984. *Production and Inventory Management.* Prentice Hall, Englewood Cliffs, NJ.

Hos, C. J., K. G. Shin. 1997. Allocation of periodic task modules with precedence and deadline constraints in distributed realtime systems. *IEEE Transactions on Computers.* **46**(12) 1338–1356.

Ji, Y., V. S. Mookerjee, S. P. Sethi. 2005. Optimal software development: A control theoretic approach. *Inf. Syst. Res.* **16**(3) 292–306.

Joglekar, N. R., D. N. Ford. 2005. Product development resource allocation with foresight. *Eur. J. Oper. Res.* **160** 72–87.

Kekre, S., N. Secomandi, E. Sönmez., K. West. 2009. Balancing risk and efficiency at a major commercial bank. *Manufacturing Service Oper. Management.* **11** 160–173.

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

Kulkarni, V. G., S. Kumar, V. S. Mookerjee, S. P. Sethi. 2009. Optimal allocation of effort to software maintenance: A queueing theory approach. *Prod. Oper. Management.* **18**(5) 506–515.

Kumar, S., Y. Ji, S. P. Sethi, D. H. Yeh. 2006. Dynamic optimization of software enhancement effort. *Proceedings of 16th Workshop on Information Technologies and Systems (WITS).* Milwaukee, WI, 133–138.

Loch, C. H., C. Terwiesch. 1998. Communication and uncertainty in concurrent engineering. *Management Science.* **44**(8) 1032–1048.

Roemer, T., R. Ahmadi, R. Wang. 2000. Time-cost trade-offs in overlapped product development. *Oper. Res.* **48**(6) 858–865.

Roemer, T., R. Ahmadi. 2004. Concurrent crashing and overlapping in product development. *Oper. Res.* **52**(4) 606–622.

Shaw, M., P. Clements. 2006. The golden age of software architecture. *IEEE Software.* **23**(2) 31–39.

Smith, R. P., S. D. Eppinger. 1997. A predictive model of sequential iteration in engineering design. *Management Sci.* **43**(8) 1104–1120.

Tavares, L. V. 1998. *Advanced Models for Project Management.* Kluwer Academic Publishers, Norwell, MA.

Online Appendix to "A Hierarchical Framework for Organizing a Software Development Process"

Appendix A: Proofs

A.1. Proof of Proposition 1

proof. It is not difficult to see that the optimal schedule is a permutation schedule; i.e., the sequence of processing forms is the same in all stages. An upper value on the completion time is obtained by inflating the processing time of each form at each stage to its maximum processing time across stages. In this case, the form with the largest inflated processing time and all the forms after it are processed in stages $2, \cdots, K$ with no inserted idle time. Therefore, the completion time is equal to the sum of the processing times plus the transition time of the form with the largest inflated processing time in stages $2, \cdots, K$, which is $\sum_{i\in\mathcal{I}} \max_{k\in\mathcal{K}}\{P_{ik}\} + (K-1)\max_{i\in\mathcal{I}}\{\max_{k\in\mathcal{K}}\{P_{ik}\}\}$. This upper value is independent of the sequence of forms and the ratio of $\sum_{i\in\mathcal{I}} \max_{k\in\mathcal{K}}\{P_{ik}\} + (K-1)\max_{i\in\mathcal{I}}\{\max_{k\in\mathcal{K}}\{P_{ik}\}\}$ to $\sum_{i\in\mathcal{I}} \max_{k\in\mathcal{K}}\{P_{ik}\}$ approaches 1 when the number of forms is sufficiently large, because K is much smaller than I. □

A.2. Proof of Proposition 2

proof. We show that MGRAM_1 and MGRAM_2 are as hard as the 3-partition problem, which is known to be strongly NP-hard (Garey and Johnson 1979). Assume that we are given a general instance of the 3-partition problem consisting of an index set $A = (1, 2, ..., 3m)$, positive elements a_i for $i = 1, 2, ..., 3m$, and a positive integer B such that $B/4 < a_i < B/2$ and $\sum_{i=1}^{3m} a_i = mB$. We now introduce a specific instance of MGRAM_1 and MGRAM_2 as follows: $K = 1$, $G = m$, $I = 3m$, $P_{ik} = a_i$ for all i and k, $D = B$, and $w_k = 1$ for all k. We shall show that the optimal solution of MGRAM_1 and MGRAM_2 takes value m if and only if the $3m$ elements of A can be partitioned into m disjoint subsets $A_1, A_2, ..., A_m$ such that $\sum_{i\in A_r} a_i = B$ for $r = 1, ..., m$. If the 3-partition problem has a solution, then it is easy to see that the elements of each subset A_r could be assigned to one of the m groups and that the total effective processing time in each group would be equal to D. In this case, each group would have one resource assigned to it: $Y_{kg} = 1$ for all k and g and

$\sum_{k\in\mathcal{K}}\sum_{g\in\mathcal{G}} Y_{kg} = m$. If the 3-partition problem does not have a solution, then there is at least one subset A_r such that the total effective processing time in that group would be greater than D. To meet the deadline, D, more than one resource would have to be assigned to this group. All other subsets would require one unit of resource, making the total resources greater than m. □

A.3. Proof of Proposition 3

proof. We shall show that the recognition version of the GM is as hard as the 3-partition problem, which is known to be strongly NP-complete. Assume that we are given a general instance of the 3-partition problem consisting of an index set $A = (1, 2, ..., 3m)$, positive elements a_i for $i = 1, 2, ..., 3m$, and a positive integer, B, such that $B/4 < a_i < B/2$ and $\sum_{i=1}^{3m} a_i = mB$. We now introduce a specific instance of the GM as follows: $G = m$, $I = 3m$, $P_i = a_i$ for all i, $R_{i_1 i_2} = 0$ for all i_1 and i_2, and $Q = B$. We shall show that the GM has a feasible solution if and only if the $3m$ elements of A can be partitioned into m disjoint subsets $A_1, A_2, ..., A_m$ such that $\sum_{i\in A_r} a_i = B$ for $r = 1, \ldots, m$. If the 3-partition problem has a solution, then it is easy to see that the elements of each subset A_r could be assigned to each of the m groups and that the total processing time of each group would be equal to B. If the 3-partition problem does not have a solution, then there is at least one subset A_r such that the total processing time of that group would be greater than B and it is easily seen that the GM has no feasible solution. □

A.4. Proof of Proposition 4

proof. We shall show that RAM_1 is as hard as the equal-size, equal-number-of-items partition problem, which is known to be binary NP-hard. Assume that we are given a general instance of the equal-size, equal-number-of-items partition problem consisting of an index set $A = (1, 2, ..., m)$, in which m is even and elements a_i for $i = 1, 2, ..., m$ are positive. Consider the following instance of RAM_1 as follows: $K = m$ and $P_{ik} = 2a_i$ if $i = k$; otherwise $P_{ik} = a_i$. Also, $D = \frac{3}{2}\sum_{r\in A} a_r$, $w_k = 1$ and $1 \le Y_k \le 2$ for all k. Finally, we consider an objective value of $D = \frac{3}{2}\sum_{r\in A} a_r$. Note that in any feasible solution to RAM_1, there exists a subset of stages, B, such that $Y_k = 2$ for all $k \in B$ and $Y_k = 1$ for all $k \notin B$. If there exists a partition, $B \subset A$, such that $\sum_{r\in B} a_r = \sum_{r\in A-B} a_r$ and

$|B| = m/2$, then setting $Y_k = 2$ for all $k \in B$ and $Y_k = 1$ for all $k \in A - B$ generates a feasible solution to RAM_1 with the objective value of $D = \frac{3}{2}\sum_{r\in A} a_r$. If no partition exists and there is a solution to RAM_1 such that its objective value is less than or equal to $D = \frac{3}{2}\sum_{r\in A} a_r$, then it is easy to see that there should exist a subset $C \subset A$ such that $|C| < |A|/2$. Setting $Y_k = 2$ for all $k \in C$ and $Y_k = 1$ for all $k \in A - C$ does not provide a feasible solution to RAM_1. □

A.5. Proof of Proposition 5

proof. We show that PLSM is as hard as the equal-size, equal-number-of-items partition problem with set $A = (1, 2, ..., m)$ and m even. Consider the following instance of PLSM as follows: $I_S = I_F = m$, $K = m$, $m_k = 3$, and $P_{ik} = 2a_i$ if $i = k$; otherwise, $P_{ik} = a_i$. Suppose the objective value is $\frac{3}{2}\sum_{r\in A} a_r$. If there exists a partition $B \subset A$, such that $\sum_{r\in B} a_r = \sum_{r\in A-B} a_r$ and $|B| = m/2$, then in each feasible solution to PLSM, each stage in the federal or state process line will use either one or two resources. Let B be the set of stages in the state process line that use two resources and let $A - B$ be the set of stages in the federal process line that use two resources. Then, the total maximum amount of work is given by $\sum_{r\in B} a_r + 2\sum_{r\in A-B} a_r$ for the state process line and $2\sum_{r\in B} a_r + \sum_{r\in A-B} a_r$ for the federal process line. Consequently, the objective value will be $\frac{3}{2}\sum_{r\in A} a_r$. Conversely, if no partition exists, we assume by contradiction that there is a solution to PLSM with the objective value of $\frac{3}{2}\sum_{r\in A} a_r$. Then, we would have:

$$\sum_{r\in B} a_r + 2\sum_{r\in A-B} a_r \le \frac{3}{2}\sum_{r\in A} a_r,$$

$$2\sum_{r\in B} a_r + \sum_{r\in A-B} a_r \le \frac{3}{2}\sum_{r\in A} a_r,$$

which lead to a contradiction. □

A.6. Proof of Proposition 6

proof. Let ϕ^f be the value of PLSM when $Y_{k_s} = \left\lfloor \widetilde{Y}_{k_s} \right\rfloor$, $Y_{k_f} = \left\lfloor \widetilde{Y}_{k_f} \right\rfloor$. Similarly, let ϕ^c be the value of PLSM when $Y_{k_s} = \left\lceil \widetilde{Y}_{k_s} \right\rceil$ and $Y_{k_f} = \left\lceil \widetilde{Y}_{k_f} \right\rceil$. Since the objective function is non-decreasing in the number of resources, we can write:

$$\frac{\phi^H}{\phi^*} \leq \frac{\phi^f}{\phi^c} \leq \frac{\sum_{i_s \in \mathcal{I}_\mathcal{S}} \max_{k\in\mathcal{K}} \left\{ P_{i_s k} / \left\lfloor \widetilde{Y}_{k_s} \right\rfloor \right\} + \sum_{i_f \in \mathcal{I}_\mathcal{F}} \max_{k\in\mathcal{K}} \left\{ P_{i_f k} / \left\lfloor \widetilde{Y}_{k_f} \right\rfloor \right\}}{\sum_{i_s \in \mathcal{I}_\mathcal{S}} \max_{k\in\mathcal{K}} \left\{ P_{i_s k} / \left\lceil \widetilde{Y}_{k_s} \right\rceil \right\} + \sum_{i_f \in \mathcal{I}_\mathcal{F}} \max_{k\in\mathcal{K}} \left\{ P_{i_f k} / \left\lceil \widetilde{Y}_{k_f} \right\rceil \right\}}.$$

For each $i_s \in \mathcal{I}_\mathcal{S}$, consider $\eta_{i_s} = \frac{\max_{k\in\mathcal{K}} \left\{ P_{i_s k} / \left\lfloor \widetilde{Y}_{k_s} \right\rfloor \right\}}{\max_{k\in\mathcal{K}} \left\{ P_{i_s k} / \left\lceil \widetilde{Y}_{k_s} \right\rceil \right\}}$ and let $k^{'}$ and $k^{''}$ be the stages that determine the maximum in the numerator and denominator, respectively. If $k^{'} = k^{''}$, then $\eta_{i_s} \leq 2$. If $k^{'} \neq k^{''}$, then $\left\lceil \widetilde{Y}_{k_s''} \right\rceil \leq P_{i_s k_s''} \left\lceil \widetilde{Y}_{k_s'} \right\rceil / P_{i_s k_s'}$. Thus, $\eta_{i_s} = P_{i_s k_s'} \left\lceil \widetilde{Y}_{k_s''} \right\rceil / \left\lfloor \widetilde{Y}_{k_s'} \right\rfloor P_{i_s k_s''} \leq 2$. The inequality also holds if we define η_{i_f} in a similar manner for federal forms. Therefore, $\phi^H / \phi^* \leq 2$. □

Appendix B: A Special Case of the Feedback Loops Approximation

Consider a process with two single-resource stages and deterministic processing times P_1 and P_2. When a form visits stage 2 for the n-th time, it returns to stage 1 for reprocessing with probability α if $n \leq \zeta$, and leaves the process with certainty if $n = \zeta + 1$, in which $1 \leq \zeta < \infty$. Therefore, if the number of times a form returns to stage 1 is r, then its processing times in the no-loops system will be $(r+1)P_1$ and $(r+1)P_2$. Let C^L_{max} and C^{NL}_{max} denote the completion time in the system with loops and in the no-loops system, respectively.

PROPOSITION B.1. *As the number of forms increases, the completion time of the no-loops process approaches the completion time of the process with loops for any realization of feedback loops; i.e.,* $\pi = \lim_{I\to\infty} \frac{C^{NL}_{max}}{C^L_{max}} = 1$.

proof. Assume that the forms are indexed according to the order of process; i.e., the first form is form 1, the second form is form 2, and so on. We first consider the case of $P_1 \geq P_2$. Let t^L denote the time when resource 1 finishes processing the last form that leaves the process with loops behind empty. Then $C^L_{max} = t^L + P_2$. Denote with t^{NL} the time when resource 1 completes processing form I in the no-loops system. Then $t^{NL} \leq t^L \leq t^{NL} + \zeta P_2$. The inequalities hold because in the system with loops, the return of forms may generate idles times for resource 2, and the total idle time is at most ζp_2. In the no-loops process, either form I does not wait for resource 2, or it must wait for resource 2 behind forms with inflated processing times. In both cases, we can write

$C^{NL}_{max} \le t^{NL} + (1+2\zeta)\,P_2$. Putting all the inequalities together and noting that $C^{NL}_{max} \ge t^{NL}$, we have $C^{L}_{max} - (1+\zeta)P_2 \le C^{NL}_{max} \le C^{L}_{max} + 2\zeta P_2$, which means $\pi = 1$.

Now, we consider the case of $P_2 > P_1$. Let φ_1 and φ_2 be the number of times forms 1 and 2 return. For forms 2,...,I, let j be the total number of forms that return and $n_1, ..., n_\zeta$ be the number of forms that return 1,...,ζ times. Clearly $\sum_{\nu=1}^{\zeta} n_\nu = j$. Also, define $\mathcal{A} = P_1 + \sum_{\nu=1}^{\zeta} n_\nu\,(\nu+1)\,P_2 + (I-j)\,P_2$. In both systems, the earliest time to complete processing all forms is when resource 2 works continuously. Therefore, $C^{L}_{max} \ge \mathcal{A}$ and $C^{NL}_{max} \ge \mathcal{A} + \varphi_1 P_1$. Since the return of a form may generate idle times at stage 1 and forms return at most ζ times, we have $C^{L}_{max} \le \mathcal{A} + \zeta P_1$. Now suppose in the no-loops system some forms generate idle times for resource 2 between processing consecutive forms. The largest idle time is generated when form 2 makes additional $\zeta - \varphi_2$ returns, because processing form 2 at stage 1 starts when stage 2 starts processing form 1. Therefore, $C^{NL}_{max} \le \mathcal{A} + (1+\varphi_1+\zeta)\,P_1 + (\zeta-\varphi_2)\,P_2$. Because C^{L}_{max} and C^{NL}_{max} are bounded by linear functions of $\mathcal{A}$, and $\mathcal{A} \longrightarrow \infty$ when $I \longrightarrow \infty$, we have $\pi = 1$. □

Appendix C: Additional Numerical Experiments

C.1. GM Lagrangian Heuristic Performance

In this section, we evaluate the quality of the Lagrangian decomposition heuristic to solving the Grouping Model (GM). We report Ave(LH/LB), which is the average ratio of the grouping penalty of the Lagrangian heuristic solution to the lower bound on the optimal grouping penalty. Note that each iteration of the subgradient method provides a heuristic solution to the GM. We use the best solution over the 35 iterations of the subgradient method. Because the solution to the GM depends on the value of δ in (13), we also vary δ.

Table 1 shows the results of the experiments. One can see that the average of Ave(LH/LB) is lowest when $\delta = 0.25$, which supports our statement that for 3 to 5 groups, the value of δ should be around 0.25. The values of Ave(LH/LB) for $\delta = 0.25$ show that the difference between the heuristic solution and the lower bound is on average 2% and has a 95% confidence interval of [1.8% 2.2%]. Note that this is a conservative estimate of the quality of the Lagrangian heuristic since we do not

have the optimal solution to the GM. Therefore, the performance of the solution procedure for the GM is quite good.

Table 1 Performance of the Lagrangian heuristic for the GM

G	K	I	Ave(LH/LB)								
			$\delta=0.10$	$\delta=0.15$	$\delta=0.20$	$\delta=0.25$	$\delta=0.30$	$\delta=0.35$	$\delta=0.40$	$\delta=0.45$	$\delta=0.50$
3	8	500	1.061	1.044	1.006	1.044	1.041	1.007	1.045	1.076	1.037
	8	750	1.020	1.032	1.031	1.009	1.015	1.028	1.027	1.011	1.071
	8	1000	1.025	1.002	1.030	1.002	1.032	1.007	1.032	1.013	1.004
	9	500	1.066	1.038	1.039	1.009	1.046	1.052	1.003	1.039	1.018
	9	750	1.027	1.020	1.009	1.043	1.053	1.008	1.033	1.023	1.002
	9	1000	1.064	1.015	1.022	1.003	1.032	1.006	1.038	1.065	1.035
	10	500	1.064	1.014	1.026	1.011	1.034	1.007	1.007	1.070	1.067
	10	750	1.005	1.057	1.026	1.017	1.030	1.022	1.041	1.040	1.021
	10	1000	1.005	1.038	1.019	1.020	1.026	1.052	1.050	1.072	1.055
4	8	500	1.043	1.021	1.048	1.018	1.036	1.050	1.041	1.029	1.053
	8	750	1.011	1.026	1.020	1.024	1.020	1.004	1.044	1.080	1.015
	8	1000	1.027	1.005	1.025	1.017	1.015	1.056	1.054	1.084	1.050
	9	500	1.069	1.034	1.020	1.018	1.013	1.017	1.056	1.067	1.055
	9	750	1.017	1.006	1.011	1.035	1.017	1.028	1.052	1.001	1.019
	9	1000	1.012	1.014	1.035	1.027	1.029	1.032	1.064	1.073	1.021
	10	500	1.049	1.022	1.013	1.021	1.048	1.018	1.032	1.052	1.009
	10	750	1.050	1.002	1.033	1.042	1.003	1.035	1.009	1.066	1.015
	10	1000	1.052	1.059	1.045	1.032	1.010	1.058	1.009	1.037	1.074
5	8	500	1.021	1.040	1.033	1.002	1.015	1.026	1.036	1.071	1.049
	8	750	1.005	1.015	1.034	1.028	1.019	1.023	1.017	1.007	1.046
	8	1000	1.019	1.002	1.038	1.008	1.000	1.011	1.007	1.000	1.007
	9	500	1.024	1.008	1.001	1.010	1.003	1.055	1.055	1.057	1.067
	9	750	1.009	1.000	1.037	1.018	1.027	1.015	1.061	1.067	1.015
	9	1000	1.075	1.009	1.011	1.007	1.003	1.003	1.030	1.008	1.052
	10	500	1.054	1.020	1.020	1.020	1.033	1.027	1.063	1.022	1.070
	10	750	1.062	1.013	1.028	1.020	1.036	1.032	1.008	1.028	1.003
	10	1000	1.008	1.023	1.030	1.041	1.025	1.044	1.052	1.022	1.028
CI (lower)			1.032	1.019	1.024	1.018	1.023	1.024	1.033	1.040	1.032
Average			1.035	1.021	1.026	1.020	1.025	1.027	1.036	1.044	1.035
CI (higher)			1.038	1.024	1.027	1.022	1.026	1.029	1.038	1.047	1.039

C.2. Computational Time of Hierarchical Models

Table 2 report the average time (in seconds) it takes to solve 90 instances of the hierarchical models for each combination of G, K, and I. The average runtime is 84.5 seconds for GRAM_1 and 47.8 seconds for GRAM_2. These computation times are quite satisfactory for tactical planning.

Appendix D: An Example of the Shortest Path Algorithm for RAM_1

Figure 1 illustrates a small example of the network construction for RAM_1. Due to limited space, we only considered five stages and did not generate all the nodes to $Y^{\max}$. Table 3 shows the

Table 2 Computation time of hierarchical models

G	K	I	Average CPU Time (seconds)	
			$GRAM_1$	$GRAM_2$
3	8	500	48.6	31.0
	8	750	66.2	31.1
	8	1000	79.7	48.0
	9	500	85.4	49.3
	9	750	87.7	53.5
	9	1000	112.8	54.5
	10	500	51.1	50.0
	10	750	58.5	61.3
	10	1000	86.2	76.4
4	8	500	73.0	33.5
	8	750	83.9	39.5
	8	1000	118.6	50.0
	9	500	77.3	41.0
	9	750	94.7	46.3
	9	1000	129.4	64.1
	10	500	65.2	45.6
	10	750	68.6	51.6
	10	1000	70.4	62.5
5	8	500	79.8	44.4
	8	750	91.2	52.3
	8	1000	106.6	55.5
	9	500	58.5	31.2
	9	750	73.0	36.8
	9	1000	100.1	41.8
	10	500	74.3	32.2
	10	750	93.3	40.3
	10	1000	110.8	49.2
CI (lower)			82.0	46.4
Average			84.5	47.8
CI (higher)			87.0	49.1

data for the example. The processing times of five forms are listed in columns two through six. The seventh column shows the cost of hiring one employee at each stage. The eighth and ninth columns show the lower and upper values on the optimal number of resources at each stage. The deadline is 20, hence $\overline{Y} = 3.85$. The shortest path is shown with dashed arcs. The optimal solution is $(3, 4, 4, 4, 4)$, with total workforce cost equal to 675.

Table 3 Data for the network example

Stage	Form 1	Form 2	Form 3	Form 4	Form 5	w_k	$Y_k^{\min}$	$Y_k^{\max}$
1	10	12	10	14	6	25	3	8
2	14	9	10	17	7	30	3	7
3	14	10	13	12	11	45	3	6
4	13	16	15	10	5	35	3	7
5	12	13	14	7	15	40	4	7

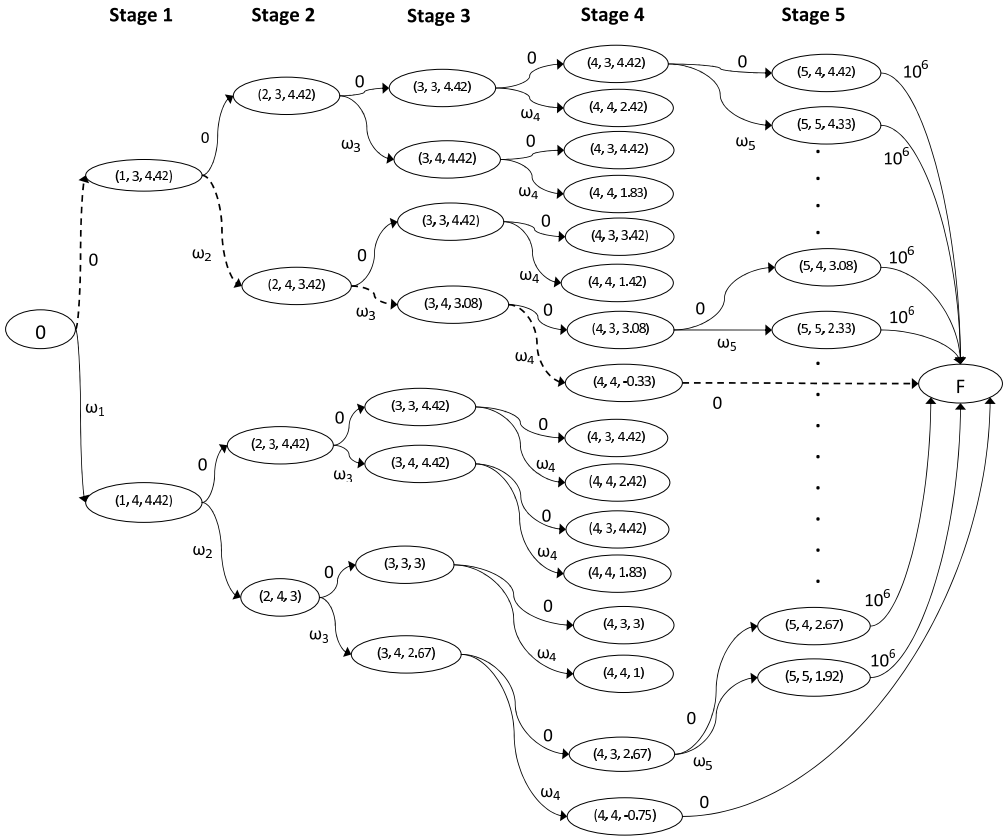

Figure 1 An example of constructing a network for RAM_1

Appendix E: Questionnaires

By way of two questionnaires, TSDC managers provided a number of the estimates we used in our models. Figure 2 shows the questionnaire for obtaining processing time distributions and percentage of forms that require rework after internal tests. Though the example presents the questionnaire for the Image Development Group (IDG), we asked the same questions for all stages. Figure 3 also shows the questions we asked for estimating the distribution of the destination of feedback loops from Integration & Final Test.

IDG Stage

	Minimum processing time (hours)	**Average** processing time (hours)	**Maximum** processing time (hours)
IDG process			
IDG internal test			

What percentage of forms have a processing time	**IDG process**	**IDG internal test**
less than 25% of the maximum processing time?		
between 25% and 50% of the maximum processing time?		
between 50% and 75% of the maximum processing time?		
above 75% of the maximum processing time?		

What percentage of forms pass the IDG internal test and do not require rework? ☐

Figure 2 Questionnaire for estimating the processing time distribution and rework probabilities

Integration & Final Test Stage

	Minimum processing time (hours)	**Average** processing time (hours)	**Maximum** processing time (hours)
Integration process			
Final test			

What percentage of forms have a processing time	**Integration process**	**Final Test**
less than 25% of the maximum processing time?		
between 25% and 50% of the maximum processing time?		
between 50% and 75% of the maximum processing time?		
above 75% of the maximum processing time?		

What percentage of forms:

pass the final test and do not require rework?	
fail the final test and return to CALC for rework?	
fail the final test and return to EF for rework?	
fail the final test and return to Interview for rework?	

Figure 3 Questionnaire for estimating the rework probability distribution at Integration & Final Test

References

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco.