Implications of Alternative Multilevel Design Methods for Design Process Management**

David Shahan, Carolyn C. Seepersad*

Abstract

Multilevel design problems are typically decomposed into a hierarchy of distributed and strongly coupled sub-problems, each solved by design teams with specialized knowledge and tools. There are two contrasting approaches to formulating and solving such collaborative design problems: (1) highly iterative exchanges of single design solutions, as in point-based optimization approaches, and (2) minimally iterative exchanges of multiple solutions, as in set-based approaches. In this paper, the effects of these alternative approaches on the overall lead time of a design process are explored. A discrete event simulation is developed to evaluate the lead times of highly iterative and minimally iterative multilevel design strategies and the sensitivity of those lead times to the level of noise in the design environment for a range of designer work loads. Designer work loads include not only the multilevel design task of interest, but also secondary design jobs that consume designer time. Noise is represented as variability in task processing times, arrival rates, and iteration levels. An example design process for an unmanned aerial vehicle is used to compare set-based and point-based design strategies. The results of the simulations indicate that the lead times of minimally iterative, set-based design processes are more robust to busy design environments than highly iterative, point-based alternatives. Accordingly, it may be advantageous to favor richer, but less frequent, exchanges of information in a multilevel design process, even if more effort is required to generate those sets of information.

Keywords:

Multilevel design exploration, Multidisciplinary design exploration, Collaborative decision-making, Design process management

** This manuscript is an expanded and revised version of a paper originally published in the proceedings of the 2007 International Conference on Engineering Design.
* Corresponding Author. Phone: (512)471-1985; Email: cseepersad@mail.utexas.edu; Mail: 1 University Station, C2200, Austin, TX 78712
1 Introduction

A multilevel design problem may be decomposed not only vertically into multiple scales or fidelity levels but also horizontally into multiple disciplines. To solve such problems, a multilevel design team decomposes a system-level problem into a set of interconnected and distributed design tasks, each solved by an individual or team of designers with very specialized knowledge and tools. These tasks often cannot be automated entirely, but require design team intervention for problem formulation, concept generation and analysis, and solution validation. The challenge is to manage interdependencies between the teams and negotiate a satisfactory system-level solution. The challenge is formidable because interactions between design teams are often complex, leading to costly iterations.

Project management techniques, such as the design structure matrix and derivative tools, are available for modeling and restructuring design processes to reduce the impact of coupling and iteration on solution lead time [1, 2]. These project management techniques manipulate the task sequences and interdependencies in order to arrive at the shortest process lead time. These techniques do not look closely at how the individual tasks are executed but characterize their behavior in high level terms that capture effects on process lead time. In contrast, design exploration techniques seek to mathematically formalize the mechanisms at work within and between each design task to arrive at the highest quality solution. These design exploration techniques can be categorized as either optimization-based or set-based approaches.

Several optimization-based approaches have been proposed for solving multidisciplinary and multilevel design problems, including analytical target cascading [3], simultaneous analysis and design [4], concurrent sub-space optimization [5, 6], collaborative optimization [7], and BLISS [8]. The difficulty with these techniques is that they require extensive iteration between design teams—one of the primary obstacles that project management tools are intended to diminish. The need for extensive iteration is driven partly by the exchange of single point solutions in optimization-based approaches, many of which are intended for fully automated execution by computers with little or no human intervention during the solution process.
Some researchers have sought to reduce iteration among design teams by exchanging richer collections of information, as part of a set-based approach for solving multilevel and multidisciplinary design problems. A set-based design philosophy has been advocated in the automotive industry [9], and set-based approaches have assumed several forms. For example, multiobjective genetic algorithms have been applied to multidisciplinary problems as a means of generating a variety of solutions for each subproblem, overcoming the convergence difficulties of gradient-based approaches, and incorporating discrete variables; however, extensive iterations between sub-problems are still required [10-14]. In a different set-based approach, robust design techniques have been utilized to generate ranged sets or intervals of design specifications that can be shared with collaborating designers [15-17]. Alternatively, some authors [18] approach the issue as a negotiation problem and formalize the negotiations by using fuzzy set theory (as part of a Method for Imprecision [19]) for modeling uncertain or imprecise parameters such as preferences for performance variables. In game theoretic approaches, design teams reduce iteration by exchanging metamodels of their individual design spaces [20-24]. In recent work [25], we have proposed a flexibility-based approach in which multilevel design teams exchange targets and Pareto sets of solutions. Like other set-based approaches, the solutions tend to be satisficing [26] or approximate solutions that are ‘good enough’ but not necessarily optimal.

Since many multilevel design tasks require active designer participation for problem formulation, concept generation, validation, and many other tasks, it is important to consider the impact of designer involvement on the efficiency of alternative design exploration approaches. Generally, computers excel at handling iterative computations in fully automated design tasks, but human designers tend to be less tolerant of high levels of iteration, and often contribute to inefficiencies in iterative design processes. When compared with the optimization-based approaches, the set- and range-based approaches tend to reduce the number of iterations between design teams. However, the set-based approaches typically require more extensive human and computing resources (per iteration) to generate a richer set of solutions. Although satisfying the demands of a richer interface may increase processing time per iteration, it may have a substantial, positive impact on the viability of the design process and the
robustness of it to variations in the human design environment. For example, design processes with higher levels of iteration may be subject to more significant queueing effects in the design process as iterative jobs compete for design team attention with other secondary jobs and interruptions such as meetings, administrative work, and work schedules. Intuitively, one would expect these delays and lead time variations to increase with the number of iterations, as the incomplete design is repeatedly passed from design team to design team. Clearly, the architecture of a multilevel design approach (e.g., optimization-based versus set-based) has an impact on the processing time per iteration and the number of iterations. *Does the architecture of a multilevel design approach also affect important metrics such as the mean lead time for the entire design process and the lead time variability? How sensitive are these metrics to the level of competing tasks in the design environment (e.g., interruptions and secondary design jobs)? Is one type of multilevel design approach more robust than another and under what conditions?*

To answer these questions, discrete event simulations are used to simulate the impact of the design environment on the performance of alternative multilevel design processes. The methodology for discrete event simulation is described in the next section, along with its relationship to relevant literature. In the third section, discrete event simulations are used to evaluate lead times for alternative set-based and point-based design processes, as applied to the design of an example unmanned aerial vehicle (UAV). Lead time means and standard deviations are investigated under a broad range of conditions, including for designer resources from other tasks and stochastic variation in arrival and completion times for each task.

2 **Discrete Event Simulation of Alternative Multilevel Design Processes**

A design process does not take place in isolation but within a design environment characterized by iteration, uncertainty, and competition for resources. Multiple projects compete for designer resources. Uncertainty surrounds the arrival time of the next project and the amount of designer time and computing resources it will require. Coupled and interdependent decisions and tasks frequently give rise to iteration and rework. All of these factors interact with design exploration strategies to influence the
efficiency and reliability with which a satisfactory solution is found; therefore, these factors are especially important for any design process model that supports comparison of set-based and point-based design processes. In this section, multilevel design processes are characterized, and a requirements list is compiled for a design process model of these processes. Based on the requirements list, a discrete event simulation tool is developed for modeling multilevel design processes and capturing the impact of point-based and set-based design exploration strategies on process performance.

2.1 Characterizing a Multilevel Design Process and Its Design Environment

A simple multilevel design process is illustrated in Figure I in which two designers solve a two-level design problem. The multilevel design project is divided accordingly into two tasks, with Multilevel Task 1 arriving in the first designer’s inbox on the left side of the figure. Once the task is completed by the first designer, it is transferred to the second designer as Multilevel Task 2. The order of the task execution, as indicated by the arrows in Figure I, implies that the downstream engineer on the right requires results from the upstream engineer on the left in order to conduct the downstream task (e.g., Designer 1 makes subsystem decisions and delivers the results to Designer 2, the system-level designer, whose system-level decisions depend on the subsystem results). If Designer 2 cannot identify a satisfactory system-wide design, iteration ensues. As noted in Section 1, two contrasting approaches to solving this problem are explored in this paper. In a point-based approach, single point solutions are exchanged iteratively between the designers until a satisfactory system-wide solution is reached. In a set-based approach, sets or batches of solutions are generated and exchanged—a strategy that typically requires longer individual task durations (for generating multiple potential solutions rather than a single one) but fewer iterations (because sets of solutions provide multiple options for subsequent designers and increase the likelihood that they can identify satisfactory overall solutions). A quantitative comparison of the expected lead times of these two approaches requires a model that considers the likely extent of iterations for each approach, the relative duration of individual set-based or point-based tasks, delays and interruptions in process flow from competing tasks and associated resource constraints, and uncertainty in all of these factors.
Resource constraints account for interactions between a multilevel design project and other tasks that vie for designer time and computing resources. When a designer receives a multilevel design task, she may be working on a backlog of tasks from other projects that have accumulated in her virtual or physical inbox, or she may be interrupted suddenly by a task that requires immediate handling. These tasks are referred to as secondary tasks or interruptions, respectively, and they are depicted entering the designers’ workspaces from the bottom of Figure I. Secondary tasks fill up the designers’ inboxes, also referred to as queues in the terminology of queueing theory. Secondary tasks may arrive if a designer is simultaneously working on multiple projects, for example. In contrast, interruptions bypass the inbox queue and immediately consume the designer’s resources. An interruption may result from emergency work requests, unexpected sick leave, meetings, or other urgent events. These secondary and interrupting tasks serve as constraints on designer resources, and resource constraints have been shown to have a significant effect on design process lead times [27, 28]. Our prior research [29] confirmed that interruptions lead to proportional increases in mean values and standard deviations of lead time. In this research, our model is focused on resource constraints in the form of secondary tasks.

Resource constraints are compounded by iterative redesign. As indicated by the return arrows in Figure I, many multilevel design tasks require iteration, either locally or globally. A global iteration occurs if the decisions made by the first designer are unacceptable to the downstream designer, and the first task must be repeated, as indicated by the return arrow at the top of Figure I. A local or internal iteration, as depicted in the smaller feedback loops, occurs as a designer repeats tasks to satisfy his own local problem constraints or goals. Global iterations between designers are subject to placement in a designer’s queue; i.e., they often wait until a designer finishes current and/or pending tasks. In contrast, local iterations imply that a designer continues to work on the current task, and the rework does not enter a queue. A common method for quantifying iteration levels is to assign a probability of iteration, which does not always provide an appropriate distribution of the number of iterations incurred by a design process, as discussed in Section 3. A more general approach is to treat the number of iterations as a
random variable that can be characterized by a probability distribution (PD). A model of a multilevel design process should include both of these mechanisms for determining the number of iterations: a probability of iteration, $p$, or a random number of iterations, $i$, drawn from an appropriate PD. In addition, local iteration can be influenced by batch processing effects associated with a set-based design process. For example, it may be necessary to iterate locally until a batch size has been reached or to change the batch size during the design process. These batch processing effects imply that the conditions for local iteration change as a result of prior events in a set-based design process, and they should be incorporated within the sequential logic of the design process model.

In addition to its role in modeling iteration, uncertainty pervades a multilevel design process and its design environment. Designers often cannot control task arrival, and task durations vary with every execution. Any model of a multilevel design process needs to be stochastic in nature to capture the effect of these variations on the overall lead time for the design process. For this study, all of the model’s arrival times and processing times are characterized by an exponential PD, which means that prior arrival times or task duration times are not indicative of subsequent arrival times or durations according to the memoryless property of the exponential distribution.

Since the phenomena discussed in this section influence the relative performance of alternative design processes, they are essential requirements for a multilevel design process model that is useful for determining when to use a point-based or a set-based design exploration method. The requirements are summarized in the next section, along with a critical review of pre-existing design process modeling efforts.

2.2 A Review of Design Process Modeling Requirements and Previous Research

The requirements for a multilevel design process model are summarized in Table I along with the capabilities of previously published approaches for project lead time estimation. A “•” in a column indicates that the factor is captured by the model used in the reference(s).

[INSERT TABLE I HERE.]
Several design process modeling techniques for lead time estimation are reviewed in Table I. Some of the most common design process models are based on the process evaluation and review technique (PERT) and the critical path method (CPM) [30, 31]. The effects of iteration and resource constraints can be quantified indirectly in PERT/CPM methods by using one’s judgment to extend task times and to shift task sequencing [31]. However, in Table I, a “•” is awarded if and only if the effect is captured directly through a programmed mechanism in the model and not through user judgment. This distinction is important when simulations are complex enough to make human judgment prone to significant errors.

Queueing network theory is another alternative for modeling design processes [32]. Some important simplifying assumptions are needed to derive closed form solutions to queueing networks—most significantly, the assumption of Markovian processes. In a Markovian process, the conditional probability of the system’s next state depends only upon the current state. This assumption prevents dynamically changing the conditions for iteration during the simulation (as required to model important set-based batch processing effects).

Discrete event simulations (DES) offer a way to generalize queueing networks to include non-Markovian effects. DES is the most general category of simulation in Table I, and several references are included to reflect the current state of evolution of DES as a design process modeling tool. Taylor et al. [33] use a DES called Q-GERT to model the design process, including a multiple project environment. However, they do not investigate the formation of queues, the use of PD’s to determine the number of iterations, or the effects of batch processing. Adler et al. [27] use a DES called SIMAN to simulate a firm’s design environment, including resource constraints, multiple projects and probabilistic iteration. They do not consider batch processing effects or use PD’s to model iteration. The research of Browning et al. [34] and Cho et al. [28] is focused on single project environments that search for the optimal

---

1 Since DES is flexible enough to include a wide variety of factors, a “•” in Table I was not used to indicate that the model was theoretically capable of capturing the factor, unless the factor was modeled specifically in the indicated reference.
sequencing and control policies for the project. While this research does not capture the impact of multiple projects, or batch processing, it is notable for its attention to the potential effect of iteration as well as the extent to which a task can be executed concurrently. Specifically, Cho et al. [28] highlight a mechanism for directly capturing the potential overlap of tasks; they also consider resource constraints. Parallel or overlapping tasks are not considered in this paper because our focus is on iteration rather than concurrency, but it is an opportunity for future research in set-based design processes, which lend themselves naturally to parallel implementations. Finally, Chen et al. [35] focus on the relationship between lead time and decision-making strategy within the design process. Game theory is used to model the decision-making strategy, and Petri nets are used to capture task flow dependencies. Petri nets include mechanisms for handling batch processing, and Chen et al. approximate the impact of batch processing on design quality, design process lead time, and designer resource requirements. However, their Petri net is not timed and hence their study does not explicitly capture these effects, relying instead on approximating formulas for lead time estimation. In contrast, this paper explicitly models the mechanisms that relate batch processing to quality, lead time, and the impact of resource constraints on lead time in a multiproject design environment.

2.3 Discrete Event Simulation of a Multilevel Design Process and Its Design Environment

Figure I depicts the discrete event simulation (DES) developed for this research. As described in Section 2.1, a multilevel design project is divided into tasks that are assigned to individual designers, and a sequence of project tasks is defined. Local and global iterations are permitted for each task. Global iterations take place between designers, and local iterations return to the same designer. As multiscale and secondary tasks arrive, they enter each designer’s inbox. A queue develops if the arrival rate of the jobs is faster than the time required to process them. Queues are capped at a maximum size, to simulate an environment in which tasks are routed to other designers or cancelled rather than accumulating in an excessively long queue. Exponential probability distributions are assigned to the arrival rates for secondary tasks and to the process (work) times for multilevel tasks and secondary tasks. For each task, the likelihood of iteration is captured with either a probability, \( p \), or a number of iterations, \( i \),
characterized as a random variable with a probability distribution. For set-based processes, batch processing effects are incorporated in the form of local iterations that continue until the required number of designs has accumulated, and global iteration probabilities (or iteration counts generated from a PD) that decrease with increasing set sizes. As described in Section 3, simulations of each design strategy are used to establish the correct multilevel task parameters for each example problem. Although this simulation-based approach may not be practical for lead time prediction by process managers, it is useful for research purposes to objectively compare the impact of alternative solution strategies on lead time.

The discrete event simulation framework developed for this research captures the cumulative effect of the factors described in sections 2.1 and 2.2, which are required for studying the differences between set-based and point-based processes with respect to the lead time of multiscale design jobs. At the beginning of the simulation, it is initialized with arrival and processing time distributions and iteration probabilities or distributions for each designer. Then, the simulation identifies the shortest time to the next event(s), subtracts this time from the remaining times for all events, and increments the simulation time. The state of each designer and his/her queue is updated according to the most recent event(s). These steps are repeated until the multilevel design job has been completed and its lead time is calculated. Since the goal is to study the lead time of the multiscale design job, the simulation starts without any multiscale design jobs (only secondary tasks and interruptions) and runs until it reaches a steady state environment at which point the multiscale design job is introduced. The multiscale design job is executed, with iterations determined by iteration probabilities or probability distributions, and the simulation ends when the multiscale design job is completed. The simulation is repeated until the mean and standard deviation of the lead time converge to within two percent.

3 Discrete Event Simulation of Alternative Design Processes for a Multilevel Example Problem

In this section, set-based and point-based strategies are used to solve a multilevel example problem, and the DES described in the previous section is used to evaluate the relative lead times of the strategies. The example is a simplified subset of the design process for an unmanned aerial vehicle
(UAV). The goal is to create a UAV that meets a predetermined requirement for cruising range, subject to a total weight constraint. Figure II provides an overview of the decomposed problem. The aerodynamics designer determines the external wing geometry as defined by the wing chord, span, angle of attack, and the NACA four digit parameterization of the airfoil cross-section and uses these values to calculate the wing’s lift, drag and pitching moment. The aerodynamics designer adjusts the span such that the total lift equals the total UAV weight and will reject wings that create a pitching moment (the twisting of the wings that points the aircraft up or down) that is too high. The aerodynamics designer will iterate locally until one or more satisfactory designs have been found. The structures designer uses the wing external geometry and loads determined by the aerodynamics designer and sets the internal wing geometry (skin thickness) such that the wing is structurally sound. If the wing can not be made strong enough, the design is rejected. Otherwise, he/she then calculates the weight of the acceptable wing. The systems designer determines the range of the UAV as a function of wing drag and weight, while finding an optimal fuselage diameter such that the range is maximized. If the final range meets or exceeds the predetermined range requirement, the design process ends, otherwise it must begin again with a new wing or set of wings from aerodynamics. The primary challenge is for the aerodynamics designer to efficiently find wing designs that not only minimize drag and satisfy moment constraints but also allow the structures and systems designers to meet structural safety factors and satisfactory range requirements. The problem is not trivial for the aerodynamics designer because it is not obvious how to identify a wing design that balances low drag with satisfactory strength, weight, and range when aerodynamics models only evaluate lift and drag. Iterative redesign often takes place.

The overall lead time for this design process depends on not only the mathematical nature of the design problem but also the solution strategy applied to it. In Section 3.1, the problem is solved with two trial-and-error solution strategies that utilize identical local search strategies and differ only in terms of their set-based or point-based coordination strategies (i.e., whether single solutions or sets of solutions are exchanged between designers). This study isolates the affect of the coordination strategy on lead time. A
subsequent study in Section 3.2 investigates design strategies that are more sophisticated and realistic but less directly comparable.

3.1 A Trial-and-Error Design Process

The trial-and-error design process is based upon randomly choosing points within the design space and sequentially evaluating them until a design is found that meets all local constraints (e.g., aerodynamics’ moment constraint and structures’ safety factor) and provides a satisfactory UAV range at the system-level. The trial-and-error process proceeds according to the flowchart in Figure II. In the point-based version of the trial-and-error process, the aerodynamics designer randomly selects a design point, evaluates it, and repeats the process until he/she finds a design that meets the aerodynamic moment constraint. This design is sent to downstream structural and system-level designers, who evaluate the design and iterate back to aerodynamics if it is not successful. If the design satisfies aerodynamic and structural constraints and meets a minimum satisfactory value for the UAV range, the design process ends, and the DES terminates. In the set-based version of the trial-and-error process, the aerodynamics designer iterates locally until he/she accumulates a batch of designs, all of which satisfy the moment constraint. Then, the entire set of designs is transferred to the downstream designers for evaluation. Since both the point-based design process and the set-based design process randomly search the aerodynamics design space, the two processes are equally efficient (or inefficient) in their design space searches. The only difference between the two processes is whether they favor local iteration (set-based) over global iteration (point-based). While these two trial and error design processes are simplistic, they provide a point of comparison between point-based and set-based design processes that is more direct than design processes that incorporate different local search strategies. Next, these two trial and error design processes are developed in greater detail.

A DES of the trial-and-error processes requires probabilities of iteration as inputs. By automating the UAV design process and repeating it, the number of times a design fails to meet a designer’s constraints can be counted, and the probability of iteration can be quantified. While this approach would not ordinarily be possible due to the computational expense of each analysis, an approximate and fast
linear vortex panel method [36] and boundary layer growth method [37] are used for rapid evaluation of the wing lift and drag. Simulating the design process 10,000 times produces the probabilities of iteration labeled as point-based in the process flow chart in Figure II. A random search that begins with $X$ wing designs will have on average $(1-p_1)X = 0.47X$ designs pass the aerodynamics moment constraint, $(1-p_1)(1-p_2)X = 0.33X$ designs subsequently pass the structural constraint, and $(1-p_1)(1-p_2)(1-p_3)X = 0.20X$ designs pass all of the constraints. For this example, a set of five trial-and-error wing designs produces one acceptable overall design, on average.

A simple set-based design process would produce a batch of results for each downstream task. This strategy decreases the probability of iterating and the expected waiting time in queue, but it increases the work time required for each designer to generate a batch of designs. For a given probability, $p$, of satisfying a design constraint, the number of iterations, $n$, required to produce $r$ successful designs follows a negative binomial distribution:

$$P\{X = n\} = \frac{(n-1)!}{(n-r)!(r-1)!} p^r (1-p)^{n-r} = \binom{n-1}{r-1} p^r (1-p)^{n-r}$$

(1)

This relationship can be used to reduce the overall probability of iterating to a desired level. For example, if designers seek a 95% chance of identifying a satisfactory design in the first iteration (and therefore not iterating between designers), they could choose set sizes such that $p_2$ and $p_3$ equal 0.025. According to the cumulative negative binomial distribution, in order to have a 0.975 probability of one design meeting the range constraint, systems needs to evaluate a batch of 4 designs. In order to have a 0.975 probability for structures to produce a batch of 4 designs that meet the safety factor constraint, aerodynamics needs to produce a set of 9 designs that meet the pitching moment constraint. If the structures group only calculates 3 or fewer good designs out of their batch of 9, then iteration occurs. However, it is not necessary for aerodynamics to produce another full set of 9 designs for structures to have a high probability of getting the additional designs needed for a full batch of 4 good designs. This subtlety requires programmed logic that changes the required batch count of the upstream task based upon the number of successful designs for the downstream task. This requirement for our DES was mentioned in
Section 2 and is now fully motivated. Using the batch generation logic built into the DES, the probability of global iteration is implicitly reduced to 0.05 (i.e., a 95% probability of identifying a satisfactory design in the first global iteration) for the DES of the set-based design process.

The DES of the point-based and set-based, trial-and-error design processes is executed with the input parameters listed in Table II. The only difference between the point-based and set-based simulations is the number of designs transferred to the downstream designers. In the point-based process, a single design is always transferred to the next designer. In the set-based process, batches of designs are transferred, according to the discussion in the preceding paragraph. The probability of a single design meeting each of the local criteria is the same for both processes, as determined by the 10,000 simulations of the point-based design process. The batch sizes for the set-based design process are chosen such that the probability of global iteration is reduced to 0.05 as discussed in the previous paragraph. These two design processes were both tested across a range of secondary task arrival times to determine the effect of resource constraints on the lead time.

[INSERT TABLE II.]

The results of the trial-and-error simulations are illustrated in Figure III, where lead time is plotted versus mean time to arrival of secondary tasks. The lead time for the set-based process is longer than that for the point-based process when secondary tasks arrive infrequently, but the set-based lead time is much shorter than the point-based lead time when secondary tasks arrive frequently. The explanation lies in the tradeoff between local and global iterations for set-based versus point-based design processes. For the UAV example, the total number of task executions is greater for the set-based process because more local iterations are needed to compile the sets of designs that reduce or eliminate global iterations between designers. Only global iterations are subject to queueing effects; therefore, secondary tasks tend to penalize global iterations and point-based processes more heavily. When secondary tasks arrive infrequently (i.e., the right side of Figure III), designers are almost entirely dedicated to the UAV design, and queueing effects are negligible. These conditions favor the point-based process because it has fewer total numbers of multilevel task executions, on average, for this example, and therefore lower lead times.
in the absence of queueing effects and wait times. As secondary jobs arrive more frequently, the point-based process is subject to increasing delays due to queueing effects. In contrast, the set-based lead time is protected from rapidly increasing lead times because it minimizes the probability of global iterations between designers. This result is even more evident in the 90 percentile lead times because the point-based process is exposed to significantly more variation from the time it spends in queue relative to the set-based process. The set-based process is less sensitive to the effects of competition for designer resources in a busy design environment in terms of both lead time mean and lead time standard deviation. In other words, a minimally iterative design process has a more robust lead time with respect to varying levels of design resource availability.

[INSERT FIGURE III HERE.]

The intersection of set-based and point-based lead times is of interest to process managers who may wish to select design strategies to minimize lead time. The exact location of the intersection depends on some of the input parameters for the DES, including arrival and work time distributions for multilevel and secondary tasks, queue capping limits, and search strategies (e.g., trial-and-error, optimization algorithms, designed experimentation). Increasing queue sizes and process variation move the intersection to the right in Figure III, favoring set-based processes. More conservative lead times, such as the 90 percentile lead times illustrated in Figure III, have the same effect. Process managers may not be able to estimate the intersection precisely and may elect to use the less risky set-based process to protect lead time from potential delays associated with inefficient exchanges between designers. Also, depending on characteristics of the example (e.g., complexity of the design space), it is possible that the point-based process will be highly iterative and time-consuming relative to the set-based process, thereby eliminating the intersection in Figure III entirely.

This simple example illustrates how a set-based design process can impact the probability of iteration and hence project lead time mean and standard deviation. The results suggest that in a design environment where designers are completely dedicated to one project, a highly iterative design process may be tenable. On the other hand, if the project requires input from groups that are dedicated to more
than one project, a set-based approach can significantly reduce lead time mean and lead time variation. The results in this section are based on a simple trial-and-error search strategy, but more sophisticated search and coordination strategies are available. In the next section, DES results are used to compare a Set-Based Method (SBM), developed by the authors, and a standard All-At-Once (AAO) optimization process.

3.2 A Set-Based Versus an All-At-Once Design Process

In this section, trial-and-error strategies are replaced with two contrasting approaches—the Set-Based Method (SBM) developed by the authors and the point-based, All-At-Once (AAO) optimization method—both of which are intended to explore the multilevel design space more efficiently than the randomized strategies employed in the trial-and-error approach.

The AAO method is equivalent to solving the design problem in a non-distributed framework. The design problem is consolidated into a single system-level design problem, such that the system-level designer selects values of all of the design variables (inputs) that satisfy all of the constraints (e.g., aerodynamics’ pitching moment constraint and structures’ safety factor constraint) and meet a satisfactory goal for the UAV system range. To solve this design problem, the system-level designer iteratively searches the integrated design space. With each iteration, the systems-level designer transfers design variable values to aerodynamics and structural designers, who execute their analysis models and transfer the results back to the systems-level designer. In this example, the aerodynamics and structural designers analyze the problem serially; aerodynamics precedes structures, because the structural analysis inputs include the aerodynamic load profile generated by the aerodynamics analysis. Accordingly, each iteration is a global iteration, requiring aerodynamics, structures, and system-level designers to serially revisit the problem.

The steps of the Set-Based Method (SBM), developed by the authors [25], are illustrated in Figure IV. In Steps 1 and 2, the design problem is decomposed, and analysis models and decisions are formulated for each designer. In Step 3, the systems-level designer identifies a set of target values for parameters that are shared between the systems-level and aerodynamics and structures. These target
values correspond to satisfactory system-level performance and may even delineate the boundaries of successful system-level designs (e.g., a maximum value for the structural weight). The goal is to decouple the systems design problem from the rest of the design process and minimize iterations between levels. In Step 4, aerodynamics and structures attempt to match the targets generated by the systems design team as closely as possible. First, aerodynamics generates a solution that meets the most ambitious wing drag target issued by the system-level designer, without regard for downstream structural performance. Since minimum drag wings tend to be very thin, it is clear that the minimum drag design is unlikely to meet structural constraints and goals. Accordingly, the aerodynamics designer generates a Pareto set of solutions that represent a compromise between matching the wing drag target issued by the system-level designer and providing a variety of wing structures that are more likely to satisfy structures constraints and goals. The structural designer receives this set of solutions and seeks to satisfy safety factor constraints while matching or exceeding the structural weight targets issued by the system-level designer. The overall result is a Pareto set of solutions that captures the trade-off of wing drag versus structural weight that is achievable by the sub-system level teams. Iteration occurs if the structural designer cannot satisfy its constraints or meet or exceed its weight targets.

[INSERT FIGURE IV HERE.]

The details of the local search processes are implemented as similarly as possible for the AAO and the SBM, to facilitate objective comparison of the two strategies. For both strategies, a sequential quadratic programming (SQP) algorithm is used as the search algorithm, as implemented by Matlab®’s \texttt{fmincon} function. (The search algorithm is used for integrated design by the systems designer in the AAO strategy and for the local search of the aerodynamics designer in the SBM strategy.) Starting points are chosen randomly. For the AAO strategy, the SQP algorithm is stopped as soon as a candidate design point satisfies all constraints and meets a minimum value for system-level performance (i.e., UAV range). Also, for the AAO strategy, if a satisfactory solution is not identified in 10 iterations, a new random starting point is chosen, and the process is repeated, to avoid excessive iteration near

\footnote{The systems design problem in the SBM is solved in 33 local iterations with the Matlab* function \texttt{lsqnonlin}.}
unsatisfactory local minima. However, the iterations associated with terminated searches are included in
the overall iteration count. These rules are implemented to make iteration levels for the AAO strategy as
comparable as possible to the SBM strategy.

With the use of random starting points for the SQP, the number of iterations required to identify a
satisfactory design point varies from starting point to starting point for both AAO and SBM searches. To
gather statistical information on this phenomenon, both processes are simulated in Matlab® 10,000 times
to generate histograms of the number of required iterations, $i_2$ and $i_3$, as well as the probability of global
iteration, $p_1$. The resulting DES network for the SBM is shown in Figure V with the local aerodynamic
iteration histogram for $i_2$ shown in Figure VI. A significant level of iteration is required by the
aerodynamics designer to generate a set of designs, as indicated by the histogram; however, these
iterations are local and are not subject to queueing delays. The batch of aerodynamics designs leads to a
0.983 probability of one local iteration and a 0.017 probability of two local iterations for the structures
designer. The set of aerodynamic designs also reduces the probability of global iteration to just 0.016 for
the SBM, minimizing the exposure of the process to queueing delays. The number of iterations for the
systems designer, $i_1$, is fixed at 33, the effort required to determine drag and weight values that achieve
the minimum satisfactory range. Once these target values are identified, the systems designer’s task is
decoupled from the remainder of the design process, as represented by the lack of a global iteration loop
in Figure V. The AAO DES network and global iteration histogram are shown in Figures VII and VIII,
respectively. Each of the global iterations of the AAO design process is subject to queueing delays. The
results for both of the strategies are presented in Figure IX.

[INSERT FIGURES V, VI, VII, VIII, AND IX HERE.]

The lead time results of Figure IX confirm the results from the previous section. While the set-
based design process typically consumes more time generating sets of designs relative to a point-based
design process, the investment is rewarded by protection from potentially drastic lead time delays
introduced during hand-offs of the job from one design group to another. The global iterations avoided
by the set-based design process introduce both delays and increased uncertainty in the delays (as
represented by the 90 percentile lead time plots in Figure IX), making the set-based design process more robust. Exactly when the lead time of the set-based design process exceeds the lead time of the point-based design process depends upon the design environment and associated DES parameters (e.g., the number of global iterations required for the AAO strategy, the processing time for secondary tasks), and typically, it is not predictable. For this example, process managers can follow the simple guideline of using a point-based design process for dedicated teams that do not incur delays from exchanging information with one another, and otherwise using a set-based design process. Of particular note for the set-based design process is the impact on lead time of the inversion of the system design team’s problem with respect to the aerodynamics and structures problems. The inversion (i.e., the system designer’s search for targets for structures and aerodynamics designers) can consume quite a bit of time although it is a decoupled activity consisting of a relatively deterministic number of iterations. In return, the inversion significantly reduces or eliminates global iterations involving the systems designer. This trade-off is fundamental to set-based design. Simply increasing the size of the set can reduce the probability of global iteration, as shown in the trial and error design processes. In the extreme, a full set can reduce the probability of global iteration to zero if it represents a complete inversion or mapping of the local design problem.

4 Discussion

Although the details of these studies, such as iteration probabilities and iteration histograms, may not be known to process managers a priori, the studies do support a simple, generally applicable result: point-based design processes exhibit more iteration than set-based design processes and hence expose the design process to the risk of longer lead times. This result is rooted in a set of fundamental, underlying mechanisms:

1) set-based design processes, when compared to point-based design processes, increase the downstream likelihood of identifying satisfactory designs and hence reduce the probability of iteration between designers,
2) increased probabilities of iteration between designers increase the sensitivity of the lead time to
delays associated with resource constraints, such as secondary tasks or interruptions that distract
designers from the project of interest, and

3) resource constraints and stochastic variations in the design environment cause not only an
increase in mean lead times but also an even greater increase in ninety percentile lead times,
eroding schedule confidence. This effect is more pronounced for highly iterative point-based
design strategies.

In Section 3, trial-and-error design processes are used to show how set-based design strategies reduce
iteration and also reduce lead time in busy design environments, relative to point-based strategies. These
trial-and-error design processes are architected to remove any distinction between solution strategies
beyond batching of design alternatives, and therefore provide an objective comparison between the two
strategies. When comparing more sophisticated design strategies that are either set-based or point-based,
this conclusion is not necessarily universally applicable. A poor strategy that uses sets may not
significantly reduce the probability of iteration relative to an excellent point-based strategy. However, if
the strategies are of comparable effectiveness, as in the comparison of the SBM and AAO strategies, the
conclusion holds that set-based strategies reduce global iteration. A comprehensive set-based strategy
that fully maps an individual designer’s design space may even eliminate the possibility of iteration.

The Set-Based Method is intended to provide guidelines for intelligently generating and
exchanging sets of designs, a subject for continued research. Another important vein of future research is
to investigate opportunities for concurrent execution of these design processes. All of the studies
presented here proceeded serially, but there is opportunity for parallelism and overlapping of tasks.
Computer-aided engineering tools with product lifecycle management and knowledge-based engineering
capabilities could help with this challenge. Finally, effective DES of set-based design processes requires
incorporating batch logic and histograms of iteration counts into the simulation. These histograms are not
easily derived without a fast surrogate design simulation that can be automated, as demonstrated in this
study. Design process managers may need to compile iteration data on real design processes and develop
and validate surrogate simulations of the design process, another rich vein of future research. It would also be interesting to compare the results of this simulation-based study to similar results from experiments with human designers in the loop.

Acknowledgments

We gratefully acknowledge support from the University of Texas at Austin, National Science Foundation grant DMI-0632766, and the Schlumberger Technology Corporation. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References


List of Figures

Figure I. The Design Environment Model for Studying the Effects of Secondary Tasks Arrivals, Interruptions and Iteration on Lead Time and Lead Time Variation
Figure II. The DES Flow Chart for the Trial and Error UAV Design Process
Figure III. The Discrete Event Simulation Results for the Trial and Error UAV Design Processes
Figure IV. The Steps of the Set-Based Design Method
Figure V. The DES Flow Chart for the SBM UAV Design Process
Figure VI. The Iteration Histogram for the SBM Aerodynamics Design Process
Figure VII. The DES Flow Chart for the AAO UAV Design Process
Figure VIII. The Global Iteration Histogram of the AAO UAV Design Process
Figure IX. The DES Results for the SBM and AAO UAV Design Processes

List of Tables

Table I. Comparison of Design Process Simulation Capabilities
Table III. Simulation Parameters for Trial-and-Error UAV Design Processes
Figure I. The Design Environment Model for Studying the Effects of Secondary Task Arrivals, Interruptions and Iteration on Lead Time and Lead Time Variation
Figure II. The DES Flow Chart for the Trial and Error UAV Design Process

Inputs: chord, span, angle of attack, NACA parameters
 Outputs: wing lift (constraint), drag (minimized), pitching moment (constraint)
Constraints: wing lift = weight limit, |pitching moment| ≤ moment limit
Model: 2D linear vortex panel model [36], boundary layer growth model [37]

Inputs: chord, span, angle of attack, NACA parameters, wing lift, wing drag, skin thickness
Outputs: wing safety factor (constraint) and weight (minimized)
Constraints: safety factor ≥ 1
Model: arbitrary cross-section beam stress calculation [38]

Inputs: wing weight, wing drag, fuselage diameter
Outputs: UAV range
Constraints: range ≥ range limit
Model: empirical drag formula [39], Brugeut range equation [40]
Figure III. The Discrete Event Simulation Results for the Trial and Error UAV Design Processes
Figure IV. The Steps of the Set-Based Design Method
Figure V. The DES Flow Chart for the SBM UAV Design Process

**Design Activity:** Generate the target design region for wing drag and weight.

**Design Activity:** Generate the set of wings exceeding or matching the drag targets and capturing the weight versus drag tradeoff.

**Design Activity:** Determine the wing weight and evaluate the set of designs against the Systems' target design region.
Aerodynamic Iterations, $i_2$

Figure VI. The Iteration Histogram for the SBM Aerodynamics Design Process
Figure VII. The DES Flow Chart for the AAO UAV Design Process
Figure VIII. The Global Iteration Histogram of the AAO UAV Design Process
Figure IX. The DES Results for the SBM and AAO UAV Design Processes
<table>
<thead>
<tr>
<th>Reference</th>
<th>Solution Method</th>
<th>Design Process and Environment Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>[32] Bolch et al. 2006</td>
<td>Queuing Network Theory</td>
<td>• • • • •</td>
</tr>
<tr>
<td>[33] Taylor et al. 1980</td>
<td>Discrete Event Simulation: Q-GERT</td>
<td>• • •</td>
</tr>
<tr>
<td>[27] Adler et al. 1995</td>
<td>Discrete Event Simulation: SIMAN</td>
<td>• • • •</td>
</tr>
<tr>
<td>[34] Browning et al. 2002</td>
<td>Discrete Event Simulation</td>
<td>•</td>
</tr>
<tr>
<td>[28] Cho et al. 2005</td>
<td>Discrete Event Simulation</td>
<td>• • • •</td>
</tr>
<tr>
<td>[35] Chen et al. 2007</td>
<td>Petri Nets</td>
<td></td>
</tr>
<tr>
<td>This paper</td>
<td>Discrete Event Simulation</td>
<td>• • • • • •</td>
</tr>
<tr>
<td>Simulation Number:</td>
<td>1: Point-based</td>
<td>2: Set-based</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Multilevel project arrival time</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>Multilevel task work time mean</td>
<td>10 for all tasks</td>
<td></td>
</tr>
<tr>
<td>Batch size</td>
<td>1</td>
<td>Variable such that $p_2 = p_3 = 0.025$</td>
</tr>
<tr>
<td>Probability of iteration, $p$</td>
<td>$p_2 = 0.29$ &amp; $p_3 = 0.39$</td>
<td>Implicitly $p_2 = p_3 = 0.025$</td>
</tr>
<tr>
<td>Secondary task work time mean</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Secondary task arrival time mean</td>
<td>1-50, 60, 70, 80, 90, 100</td>
<td></td>
</tr>
<tr>
<td>Maximum queue length</td>
<td>9 (10 in the system including the job being worked)</td>
<td></td>
</tr>
</tbody>
</table>