Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2016), *30*, 107–121.

doi:10.1017/S0890060415000311

A shortest path method for sequential change propagations in complex engineering design processes

YULIANG LI,[1] WEI ZHAO,[2] AND YONGSHENG MA[3]

[1]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
[2]Department of Foreign Language, Zhejiang University of Finance and Economics, Hangzhou, China
[3]Department of Mechanical Engineering, University of Alberta, Edmonton, Alberta, Canada

(RECEIVED June 7, 2014; ACCEPTED March 1, 2015)

Abstract

Engineering design changes constantly occur in complex engineering design processes. Designers need appropriate measures to handle the numerous design changes in order to realize consistent and completely validated product models so that successful product development is assured. In this paper, a time-based mathematic model is presented to characterize the sequential change propagation process, and then the shortest path algorithm is given to find the most timesaving routes for changes to propagate to other dependent design tasks. An analysis method is introduced to compute the sensitivities of change impacts on the affected design tasks, which indicates that the more time consumed by a change to take its effect, the more sensitive the change impacts on those downstream dependent tasks. A case study of change propagations in motorcycle engine design process was presented to demonstrate the proposed method.

Keywords: Change Propagation; Design Process; Sensitivity Analysis; Shortest Path Method

1. INTRODUCTION

Design change is defined as an active revisit of a task that has been considered completed (Jarratt et al., 2011). Many sources can cause design changes in a product development process, for example, new customer requirements, technology innovations, problems with past design, and so on (Eckert at al., 2004). Because interdependent relationships commonly exist among design tasks, changes to one task are likely to lead to changes to another one, which in turn can propagate further. During the propagation process, design changes can generate significant impacts on product design time and cost, so they should be carefully managed to avoid unnecessary and overly negative influences on the product development process.

1.1. Review of change propagation analysis and prediction methods

Design change propagation analysis and predictions on the impacts of product development time and cost are important engineering processes to realize successful innovation. In the literature, many references can be found related to analyzing change propagations and predicting change impacts on the product development process (Clarkson et al., 2004; Eckert et al., 2004; Giffin et al., 2009; Lee et al., 2010; Jarratt et al., 2011; Siddiqi et al., 2011; Yang & Duan, 2012; Hamraz et al., 2013*a*, 2013*b*). Eckert et al. (2004) made change analysis based on a detailed case study in a helicopter company. In their study, change types, reasons, change propagation patterns, and so on, were identified. Clarkson et al. (2004) proposed a risk-based change prediction method, where design efforts would be directed toward low-cost subsystems according to the distribution of resulting risks for different propagation paths. Lee et al. (2010) introduced an analytic network process to measure design change impacts in modular product. Hamraz et al. (2013*a*) introduced a matrix-calculation-based algorithm for numerical change propagation analysis to account for the exclusion of self-dependence and cyclic propagation paths. Yang and Duan (2012) proposed a method for searching change propagation paths with a study of parameter linkages, but clearly the acquisition of an optimal change solution imposes a significant challenge for designers due to the nonuniqueness of change propagation paths. In the change propagation analysis reports given by Giffin et al. (2009) and Siddiqi et al. (2011), concepts such as frequency of change patterns and strength of product components on the absorber–multiplier spectrum were introduced, and a multidimensional approach was suggested to identify the best design and management strategies among similar systems and

Reprint requests to: Yuliang Li, State Key Lab of CAD&CG, Zhejiang University, 38 Zheda Road, Hangzhou, Zhejiang Province 310027, China. E-mail: lyl_zw@zju.edu.cn

projects. For more literature on change prediction and analysis methods, please refer to the review reports given by Jarratt et al. (2011) and Hamraz et al. (2013*b*). It should be pointed out that in these studies generally local-impact based evaluation methods are used to select change propagation paths; although all the possible paths that changes can propagate along from the initial task or component to other ones are taken into account to calculate the impact, the selection results may not be optimal in the global perspective because the impact on the whole design process or product is not considered. To make up for this deficiency, an overall-impact based method is needed to find the optimal solutions for changes in the complex engineering design process.

1.2. Review on activity criticality analyses

Another issue related to change predictions is activity (task) criticality analysis. In engineering design process management, frequent changes to critical design tasks or critical product components should be avoided. In this field, Browing and Eppinger (2002) took the interface criticality index as the evaluation standard to analyze the relative importance of interfaces among activities. However, the sensitivity of change impacts on the product development process was not computed. To help manage the simultaneous execution of coupled design tasks in concurrent product development, Krishnan (1996) proposed the concept of "high" and "low" *downstream sensitivity* without giving a quantitative measure. In the project management field, a few methods have been proposed to measure the criticality indices of the activities and paths via stochastic activity networks (Dodin & Elmaghraby, 1985; Bowman, 1995; Mummolo, 1997), but they cannot be applied to looped activity networks. In this paper, to help designers make decisions on change propagations, a local change sensitivity computation method is proposed to quantitatively measure the criticality of changes.

1.3. Research motivations and structure of the paper

In this paper, a shortest path method based on change propagation simulations is presented to find the solutions for the sequential change propagations. Note that in the graph theory, the *shortest path* refers to a path with the shortest length between any two vertices in a graph, but in this paper, the shortest path is defined as a path between any two task vertices in a design process network that can propagate the change and fulfill the change requirement with the minimal impacts. In this definition, the vertices and edges contained in the paths are neither explicit nor distinct due to design iterations, which differ with the definition of the shortest path in the graph theory. The proposed method addresses the following two issues:

1. Given a design change in a highly interconnected product development process, what is the solution that has the minimum time cost of rework caused by the change if the affected design tasks are solved serially?
2. How many parts or design tasks would be affected by the change if the solution is implemented, especially when the magnitude of change is not constant?

These two questions are important, especially when designers are pressed to deliver their jobs and yet have to satisfy the change requests quickly for a feasible solution. The proposed method is expected to facilitate the designers in making the decisions based on the analyses of the above two issues created by changes and, hence, contributes to the research field where currently, predicting design changes with changing magnitudes is basically ignored in the global perspective.

To implement this proposed method, a digraph-based model is adopted to represent the product design process and output logical dependence relationships are incorporated into the process model, in which all the optional tasks and paths that design changes may evolve along are included, although only a subset of tasks and paths is to be traversed when a specific design change is to be fulfilled. Based on the model, the shortest path method combined with change propagation simulations (Wynn et al., 2010; Li et al., 2012) is searched for the most timesaving change propagation route in order to obtain the shortest process time for fulfilling design requirements. In addition to change durations or costs, a quantitative sensitivity computation method is given to analyze the criticality of different change propagation paths. This change sensitivity measure can assist designers in the selection of change paths and determination of concurrent execution of upstream and downstream design tasks.

The remainder of the paper is organized as follows: Section 2 introduces the process model, Section 3 describes the shortest path approaches, Section 4 gives the sensitivity analysis method, Section 5 presents a case study, and Section 6 discusses conclusions and future work.

2. MODELING SEQUENTIAL CHANGE PROPAGATIONS

Observed from engineering design practice and as demonstrated in the literature (Clarkson et al., 2004; Yang & Duan, 2012), many design change paths are available, representing different solutions for a change requirement, especially when the requirement can be satisfied by changing different product components with the similar effect. These solutions may be independent to each other or interdependent. Take the design of a reducer as an example to demonstrate the concept (see Fig. 1a). Suppose the total weight or mass has to be reduced in the overall design task. This requirement can be addressed by different design tasks, for example, the reducer cover design task, reducer body design task, gear shaft 1 design task, and so on. Certainly different design impacts will be incurred due to the above design tasks if different solutions are implemented. Considering these design tasks have dependence relationships with design tasks of other components in the product (Fig. 1b), their change results may possibly affect those tasks too, and thus the total

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

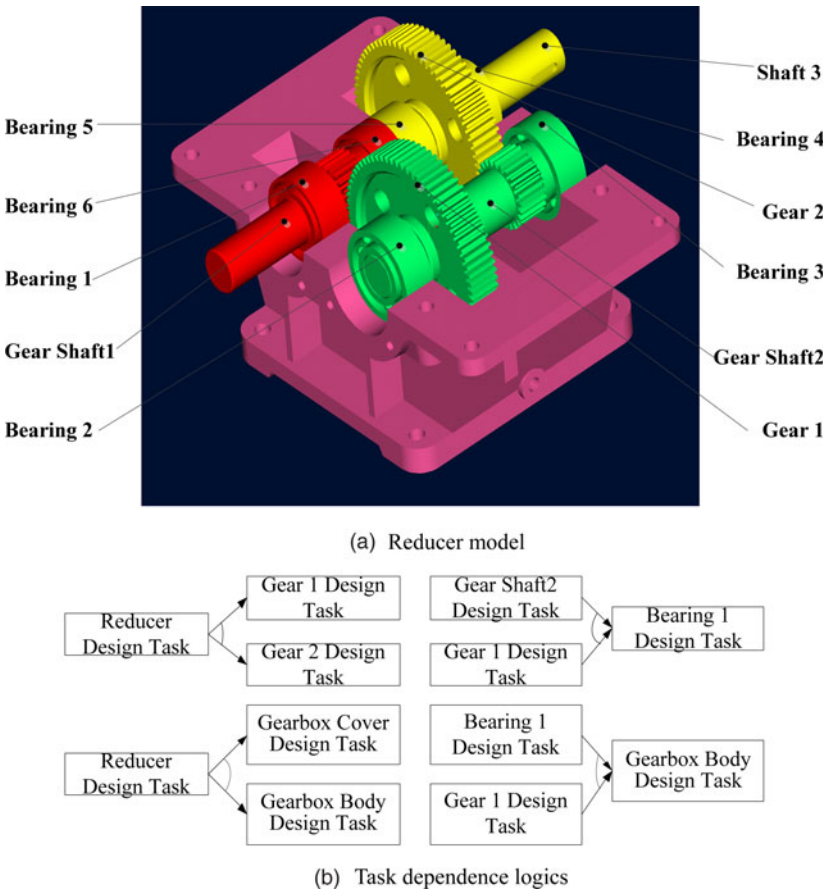

Fig. 1. Reducer model and some task-dependence logics.

design process may be impacted differently by different selections of change evolution routes. For example, the total engineering time or cost, or eventually the manufacturing cost, will be different for completely fulfilling the initial change requirement. Therefore, the selection criteria for the change propagation path should be based on not only the initial change magnitude but also the change impacts on the whole engineering and manufacturing process.

To model the change propagation process, design task and task dependency are two basic elements. Design task is defined as to apply a designer's scientific and engineering knowledge to the solution of technical problems (Pahl et al., 2007). Design task is a general concept that spans all the product development process, for example, conceptual design task, structural design task, tolerance design task, and so on. In this paper, design task mainly refers to the structural design because it is one of the most important design tasks in mechanical product development processes. However, the task dependency includes not only the geometric or structural dependency between parts but also the necessary information for solving structural design tasks, for example, material, stress, velocity, and so on. Therefore, the model and the method presented in this paper can be applied to design tasks in other product design stages as well. Task dependencies can have different logic relationships in addition to their different contents. In this paper, three kinds of output patterns, namely, Split-Or (Fig. 2a),

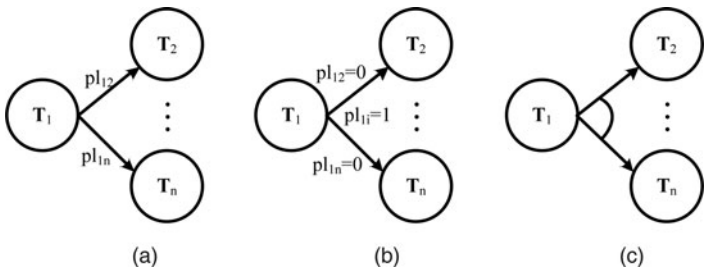

Fig. 2. Output logics for modeling design process.

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

Split-Xor (Fig. 2b), and Split-And (Fig. 2c), are taken into consideration in the change process model. The Split-And disjunction means all the downstream tasks emanating from the change task node are affected by the change simultaneously, while some or only one branch downstream tasks are affected by the instigating change respectively for the Split-Or and Split-Xor disjunctions. Theoretically, the adjacent downstream tasks emanating from the Split-And disjunction node can be merged into the split task node to constitute a new task because they are closely related when design changes occur.

When there are Split-Or and Split-Xor disjunctions in the design process, the parallel and sequential change propagation patterns are available, as shown in Figure 3. For the parallel change propagation pattern, the downstream tasks originating from the same upstream design task are executed concurrently if they do not have dependence relationships (Fig. 3a). For the sequential change propagation pattern, each design task in the queue is solved one after another (Fig. 3b). It can be seen that the parallel change propagations are composed of several sequential change propagations. When there are Split-Or and Split-Xor disjunctions that represent different change propagation paths in the design process model, an issue arises as to how to select one path from the potential solutions that can satisfy the initial change requirement with the minimum change cost.

To model the sequential change propagation, two variables, namely, propagation likelihood pl_{ij} and propagation impact pi_{ij}, are proposed to represent the information transferred between two directly dependent design tasks T_i and T_j. Propagation likelihood pl_{ij} is the proportional distribution of the change results in task T_i to the downstream task T_j, propagation impact pi_{ij} is the rework proportion of task T_j caused by the upstream task T_i with the change propagation likelihood pl_{ij}. In the paper, pi_{ij0} is used to describe the largest rework proportion of the task when the upstream task transfers the change to the downstream one with the maximal propagation probability, and pi_{0i} is used to represent the initial change impact (ICI) caused by the initial or emergent change to task T_i. Suppose a design task T_i transfers a design change to task T_j with the propagation likelihood of pl_{ij}, and the interval of propagation likelihood from task T_i to task T_j is $[pl_{ijl}, pl_{iju}]$,

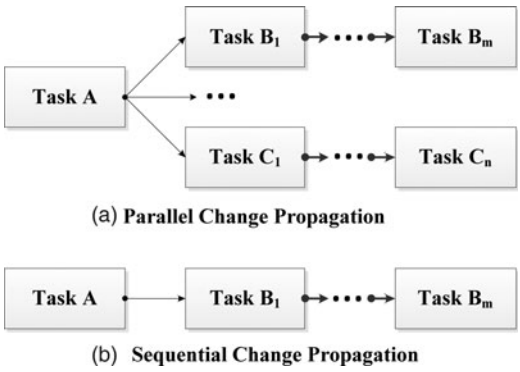

Fig. 3. Parallel and sequential change propagations.

then the propagation impact for the kth iteration, taking the learning (curve) effect into consideration, is

$$pi_{ij}(k) = \frac{pl_{ij}(k) - pl_{ijl}}{pl_{iju} - pl_{ijl}} \times pi_{ij0}^{k}. \tag{1}$$

The equation is derived based on the assumption that the propagation impact has a linear relationship with the propagation likelihood. In the equation pi_{ij0}^{k}, namely, the k**th** power of pi_{ij0}, or the k**th** order of design iterations from task $\mathbf{T}_i$ to task $\mathbf{T}_j$, is the learning effect produced by designers as they become more and more familiar with design tasks. Because pi_{ij0} is usually less than 1, the power of pi_{ij0} represents a decreasing impact in the later design iterations, which also means the designer spends less time on handling the change. Certainly the equation is also applicable to the situation that pi_{ij0} is greater than 1. The above equation holds when the lower bound of the propagation likelihood is 0. However, when the lower bound is not zero, that is, when the two tasks have some inherent dependent relationships, the following equation can be used,

$$\begin{cases} pi_{ij}(k) = \dfrac{pl_{ij}(k) - pl_{ijl}}{pl_{iju} - pl_{ijl}} \times (pi_{iju} - pi_{ijl}) + pi_{ijl}^{k} \quad pl_{ij}(k) \geq pl_{ijl} \\ pi_{ij}(k) = pi_{ijl}^{k} \quad pl_{ij}(k) < pl_{ijl}. \end{cases} \tag{2}$$

Here pi_{ijl} and pi_{iju} are the lower and upper bounds of the propagation impacts the task $\mathbf{T}_i$ can create on task $\mathbf{T}_j$, and pi_{ijl}^{k} is the k**th** power of pi_{ijl}. In this paper, to make the process model easy to use, Eq. (1) is adopted to model the propagation impacts among design tasks because sometimes it is difficult to elicit explicitly from designers the impacts a design change can cause according to our interview with designers in industry, let alone the upper and lower impact bounds.

In the change propagation process, when a design task is affected by some upstream task to a little extent, the change results will have a low possibility to affect its downstream task, and vice versa. Therefore, the propagation likelihood should have a relation to the impacts the task receives. Specifically in the kth design change iteration, the propagation likelihood of change effect on a downstream task $\mathbf{T}_j$ created by a design task $\mathbf{T}_i$ is proportional to the rework amount of the task $\mathbf{T}_i$, namely,

$$pl_{ij}(k) = pl_{ij} \times pi_{i-1,i}(k). \tag{3}$$

Here $pi_{i-1,i}(k)$ is the impact some upstream task $\mathbf{T}_{i-1}$ imposes on task $\mathbf{T}_i$. The total duration for completely resolving an initial or emergent design change is

$$D(C) = \sum_{i=1}^{n} pi_{i-1,i} \times D_{T_i}. \tag{4}$$

Here n is the number of changes caused by the initial or emergent change (including the instigating one), and D_{T_i} is the duration for the task T_i where change $C_i (i = 1, \ldots, n)$ occurs.

Take the simple process model shown in Figure 4 as an example to demonstrate the above concepts. The propagation

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

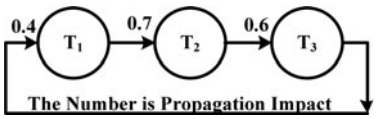

Fig. 4. A simple change process model.

impacts between any two tasks are shown in the figure, and the upper and lower bounds of the propagation likelihoods are all 1 and 0, respectively. The task durations for the three tasks are 6, 4, and 7 days. Suppose an emergent design change happens in task T_1, and 20% of the original design work needs to be redone to satisfy the change requirement, namely, the ICI pi_{01} is 0.2, then according to the Eqs. (1) and (3), we have

$$pl_{12}(1) = pl_{12}(0) \times pi_{01} = 1 \times 0.2 = 0.2,$$
$$pi_{12}(1) = \frac{pl_{12}(1) - pl_{12l}}{pl_{12u} - pl_{12l}} \times pi^1_{120} = \frac{0.2 - 0}{1 - 0} \times 0.7 = 0.14,$$
$$pl_{23}(1) = pl_{23}(0) \times pi_{12} = 1 \times 0.14 = 0.14,$$
$$pi_{23}(1) = \frac{pl_{23}(1) - pl_{23l}}{pl_{23u} - pl_{23l}} \times pi^1_{230} = \frac{0.14 - 0}{1 - 0} \times 0.6 = 0.084,$$
$$pl_{31}(1) = pl_{31}(0) \times pi_{23} = 1 \times 0.084 = 0.084,$$
$$pi_{31}(1) = \frac{pl_{31}(1) - pl_{31l}}{pl_{31u} - pl_{31l}} \times pi^1_{310} = \frac{0.084 - 0}{1 - 0} \times 0.4 = 0.0336,$$
$$pl_{12}(2) = pl_{12}(0) \times pi_{31} = 1 \times 0.0336 = 0.0336,$$
$$pi_{12}(2) = \frac{pl_{12}(2) - pl_{12l}}{pl_{12u} - pl_{12l}} \times pi^2_{120} = \frac{0.0336 - 0}{1 - 0} \times 0.7^2 = 0.016464,$$
$$pl_{23}(2) = pl_{23}(0) \times pi_{12} = 1 \times 0.0.016464 = 0.016464,$$
$$pi_{23}(2) = \frac{pl_{23}(2) - pl_{23l}}{pl_{23u} - pl_{23l}} \times pi^2_{230} = \frac{0.016464 - 0}{1 - 0} \times 0.6^2 = 0.00592704.$$

The remainder change impacts are omitted because they are very small. Then the total duration for resolving the change is

$$D(C) = 0.2 \times 6 + 0.14 \times 4 + 0.084 \times 7 + 0.0336 \times 6 + 0.016464 \times 4 + 0.00592704 \times 7 = 2.66(\textbf{Day}).$$

3. THE SHORTEST PATH METHOD FOR SEQUENTIAL CHANGE PROPAGATION

In light of the mathematic model and output patterns, the shortest path algorithm can be presented to find the most timesaving change propagation paths. However, one issue regarding design iterations must be resolved before the algorithm is introduced. In the literature, it is argued that the change propagation process should not contain design iterations because design changes occur when designs are released, and this was regarded as an important difference between change design and routine design (Jarratt et al., 2011). However, because iteration is the nature of design (Eppinger et al., 1994), it cannot be excluded that iterations will not happen in the change design process. Generally, design iteration refers to the application of design actions to the entities at different levels with the same product or set of products to correct a discovered discrepancy or other variation from requirements (Bhise, 2014). In the change propagation process, it is emphasized that changes to one part or system can have a knock-on effect on other parts or systems (Eckert et al., 2004). Thus, it is highly possible that change iterations can appear in the change propagation process, especially when multiple input or requirement discrepancies are propagated to the same entities at different times. Change iterations did occur in complex engineering design process, as reported by Giffin et al. (2009). Hamraz et al. (2013*a*) assumed that the design change propagation process should not contain cycles on the project management level because the efforts of small loops were already taken into consideration when designers were asked to estimate change efforts. In this paper, change iterations are included in the change propagation process due to the level of task granularity is low, task-based data are available for modeling propagation process; as will be seen later in the paper, when there are complicated linkages between tasks, it is highly possible for a change to a task to affect the task again through a number of intermediate steps.

Based on the above assumptions, a change propagation path is defined as a finite sequence of task vertices and directed dependence edges of the process network

$$\mathbf{T}_i\,\mathbf{e}_1\,\mathbf{T}_{p_1}\,\mathbf{e}_2\,\mathbf{T}_{p_2}\,\mathbf{e}_l\,\mathbf{T}_{p_k}\cdots\mathbf{T}_{p_m}\,\mathbf{e}_n\,\mathbf{T}_j \quad (l = 1, 2, \ldots, n;\ k = 1, 2, \ldots, m, i, j), \quad (5)$$

where the start and end nodes are the change instigating and stop tasks $\mathbf{T}_i$ and $\mathbf{T}_j$, respectively, which may be any two same or different task nodes in the process network, m represents the number of distinct tasks except the instigating and stop tasks, n is the number of edges or dependencies contained in this path, and an edge or dependency may appear twice or more in the path. According to this definition, it can be seen that a change propagation path is not really a graph path defined in the graph theory (Wilson, 1996) because the vertices and edges in the sequence are not distinct. However, the term *path* is still used in the paper to represent the change propagation routes in a more vivid way. Based on this definition, the shortest change propagation path is defined as follows: given a directed and cycled task network and an instigating change task T_i with an ICI pi_{0i}, find all the shortest change propagation paths that the change can propagate to other tasks with the greatest change propagation likelihood until it cannot generate significant impacts on them. Here the ICI refers to the initial or emergent change impact on the instigating design task, and the significant impact, defined as greater than or equal to 1% of the original design effort needed to solve the task, is taken as the stop criterion for change propagations. Certainly different threshold values can be assigned to different design tasks as the propagation stop criterion. Due to this threshold value, a change propagation path may be a solution for design changes within an interval

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

of ICIs, and the intervals for two different propagation paths may have an intersection; that is, for two changes with different ICIs, the same change propagation path may be found.

Based on the above definition, the shortest change propagation path problem and the shortest path problem in the digraph theory differ in the following points: first, the process network through which the change propagates usually contains cycles and the length of path is determined not only by the initial change magnitude and the network topology but also by the decreasing rate of impacts the change may create on the downstream tasks; second, not all task nodes in the process network may be involved in the change propagation paths, and some vertices and edges may appear in a path twice or more, and this may make the number of shortest change propagation paths much larger than the number of shortest paths in the same process network model; and third, the change propagation processes must be recorded in order to output the shortest propagation paths from the original task node to other nodes. For this last point, a simulation-based algorithm for finding the shortest change propagation paths is presented, as shown in Figure 5, among which Figure 5a is the main procedure and Figure 5b is a subroutine for creating a simulation step to handle the change impact.

The algorithm is obtained by modifying the breadth-first-search (BFS) based graph traversal method, which is similar to the Dijkstra algorithm for finding the shortest paths from a single source vertex to all other vertices in a graph (Dijkstra, 1959). Considering that the algorithm is too long, it is divided

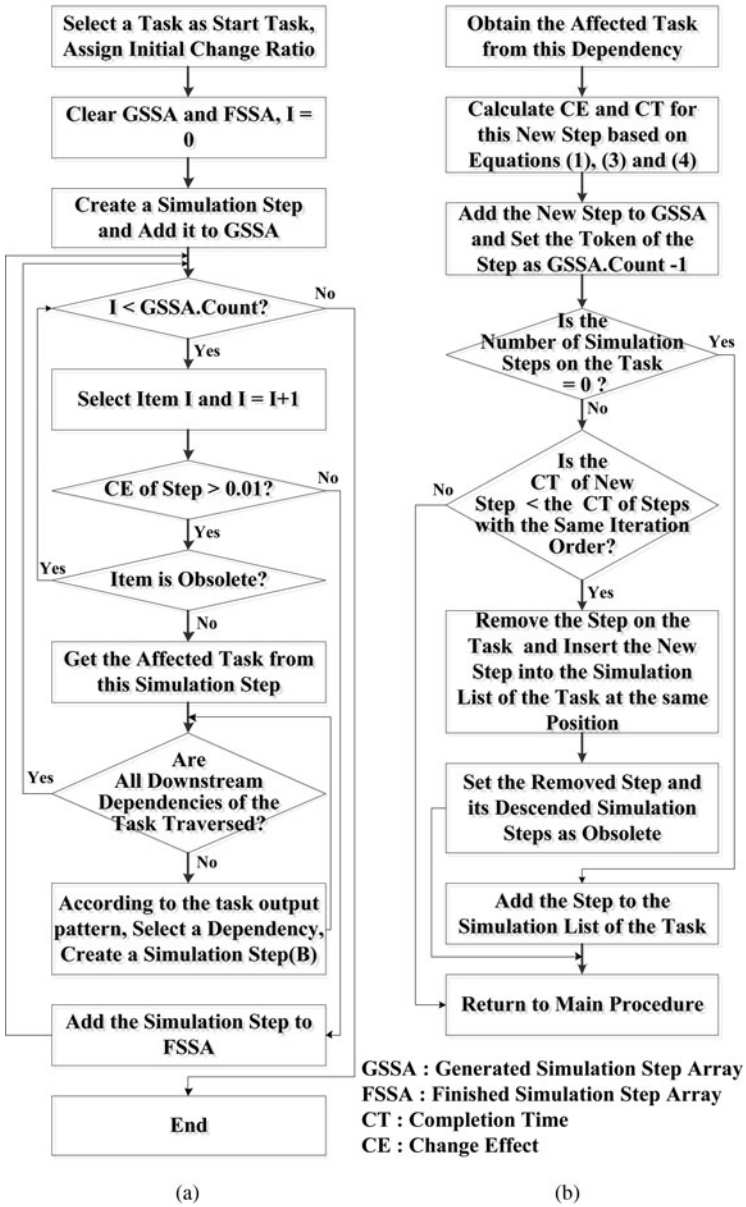

Fig. 5. The shortest change propagation path method.

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

into two parts: a main procedure (Fig. 5a) and a subroutine (Fig. 5b). Modifications to the standard BFS algorithm include when changes leave the first-in–first-out queue, only changes with their change effects greater than the criterion, namely, 1% of the original design effort, will propagate to downstream tasks; otherwise, they will be regarded as finished changes. Thus, whether a change to a design task can propagate to other adjacent tasks or not is not determined by whether the task node is visited, as in the standard BFS algorithm, but the magnitude of its change effect. This may lead to multiple visits of the same task node in the simulation process. In the generation of a simulation step, as shown in the subroutine of Figure 5, the new change will affect the existing changes to the same task node according to the shortest path principle if their iteration orders are equal. This is different with the simple visit of a graph node in the BFS algorithm.

In the algorithm, three data arrays are used to record the traversal process, namely, generated simulation step array, finished simulation step array (FSSA), and an array of simulation steps created for each design task. Generated simulation step array is a first-in–first-out queue, and is used to record each change simulation step generated in the change propagation process. It can stop the simulation algorithm when the array is empty. FSSA is used to record those change steps with their change effects less than the propagation criterion; that is, changes in the FSSA will not propagate to other design tasks any more. According to the relationships between simulation steps, the records in the FSSA can be used to find out all the shortest paths along which the initial or emergent design change can propagate from the initial design task to other tasks. For the task simulation array, if the completion time of the later generated simulation step is earlier than the former steps with the same iteration order, then the former step will be removed from the array and the state of the step and its descended simulation steps set as obsolete, so that no new steps will be generated from them. The array for each task stores the simulation steps created on the task by the fastest change for each iteration order. When the rework of the change impact or effect is less than 1% of the original design task, the change will never propagate. All the shortest paths from the source task node to other task nodes can be tracked back according to the sequential relationships among the simulation steps.

As for the algorithm itself, when it is compared to the existing shortest-path based algorithms, such as the Viteri (1967) algorithm, our algorithm has to handle the following more complicated scenarios. First, the Viteri algorithm is used to find the most likely sequence of finite states or events; there is no stop criterion and no iteration in the algorithm. Second, in the Viteri algorithm, the number of "propagation" steps is deterministic and all the potential states move forward simultaneously; it does not need to deal with impact differences that constantly occur in the change propagation process; namely, later changes may have bigger impacts than earlier ones. As far as the change prediction method is concerned, our method has the following advantages compared with existing ones (Clarkson et al., 2004; Lee et al., 2010; Yang & Duan, 2012; Hamraz et al., 2013*a*):

1. Our method can find the most timesaving change propagation path according to the change impact on the whole product development process rather than the impact on some local design task in the process. The local-impact based selection criterion for directing change propagations in existing methods may not be optimal in terms of global impact.
2. Our method can quantitatively predict how many design changes and what design tasks may be involved in a design change propagation process, and this can facilitate designers' comprehension of the whole change process and preparation for design actions to completely resolve the initiated change.

Certainly, our method also suffers from some limitations; for example, design dependencies between design tasks or components are simply modeled as two numbers, namely, propagation likelihood and propagation impact, and more complicated constraints are not taken into consideration, which may lead to a large amount of shortest change propagation paths given an initial or emergent change with a big ICI. This is not true according to our survey in industry, and certainly merits further research work in the future.

4. SENSITIVITY ANALYSIS OF CHANGE PROPAGATION IMPACTS

In the program evaluation review technique or critical path method based project network scheduling problems, the concept of critical path is often used to refer to the sequence of activities whose accomplishment will require the greatest time (Kerzner, 2009). This concept is useful in evaluating the product development process, but in terms of design changes, in addition to the change duration, the sensitivity of impact created on downstream tasks by the change of upstream tasks is an important evaluation index for designers to select the change propagation paths. The more sensitive the impact created on a change propagation path is with respect to a design change, the more efforts designers must spend on resolving the same amount of changes because except for a few special cases, the sensitivity function for product change process is usually nondecreasing (Krishnan, 1996). Thus, in different product development phases, the sensitivity index can be used to help designers make design decisions. For example, at the early stage, in order to speed up the design evolution process, upstream and downstream design tasks can be overlapped if the downstream sensitivity value is not big; while at the middle product development stage, less sensitive change propagation paths are expected to be selected by designers in order to implement solutions to changes with less effort; at the later development stage, design change requests may be rejected if the sensitivity index is big and designers are pressed for delivering their jobs.

https://doi.org/10.1017/S0890604415000311 Published online by Cambridge University Press

As stated in the paper, different solutions may be found for resolving a change requirement, but designers may spend different efforts for these solutions, and these efforts are related to the specific solution selected by the designer and the ICI. Therefore, the sensitivity of the total change propagation duration for a particular path with respect to the instigating change impact is presented to measure the impact sensitivity for the solution represented by the path. According to the definition of sensitivity of propagation path and the change propagation process model, the sensitivity index can be calculated as follows:

$$SS_i = \frac{\begin{array}{r}(pi_{0,1} + \Delta pi_{0,1}) \times D_1 + \\ \sum_{j=2}^{n_i} \left[(pi_{0,1} + \Delta pi_{0,1}) \times \left(\prod_{k=2}^{j} pi_{k-1,k} \right) \times D_j \right] \\ - \sum_{j=1}^{n_i} \left[\left(\prod_{k=1}^{j} (pi_{k-1,k}) \right) \times D_j \right]\end{array}}{\Delta PI_1}$$

$$= \frac{\begin{array}{r}(pi_{0,1} + \Delta pi_{0,1}) \times \left\{ D_1 + \sum_{j=2}^{n_i} \left[\left(\prod_{k=2}^{j} (pi_{k-1,k}) \right) \times D_j \right] \right\} \\ - pi_{0,1} \times \left\{ D_1 + \sum_{j=2}^{n_i} \left[\left(\prod_{k=2}^{j} (pi_{k-1,k}) \right) \times D_j \right] \right\}\end{array}}{\Delta pi_{0,1}}$$

$$= D_1 + \sum_{j=2}^{n_i} \left[\left(\prod_{k=2}^{j} (pi_{k-1,k}) \right) \times D_j \right], \quad (6)$$

where D_j is the duration for solving task T_j ($j = 1, \ldots, n_i$), n_i is the number of tasks in the sequential task block, $pi_{0,1}$ is the given ICI on the instigating task $\mathbf{T}_i$, $\Delta pi_{0,1}$ is a small increment of the ICI, $pi_{k-1,k}$ is the change effect from task $\mathbf{T}_{k-1}$ to $\mathbf{T}_k$, and

$$\prod_{k=2}^{j} (pi_{k-1,k})$$

is the change impact transferred from task T_1 to T_j. In the equation, the former part is used to calculate the total change duration after a small increment of change impact is added to the given ICI, while the later part calculates the change duration with the given ICI. Although the ICI $pi_{0,1}$ does not appear in this equation, that does not mean the sensitivity is not relevant to the initial impact because, first, different ICIs may affect different number of tasks, namely, n_i in the equation, so the sensitivity is also related to the initial impact; and second, when the change impact is less than 1% of the original design effort, the change will not propagate any more, and the design impact will not change although the ICI may change. Due to this interception error caused by the change propagation threshold value, there are some ranges in which the impact sensitivity will not change with respect to the ICI; in this case, it seems the sensitivity is irrelevant to the ICI.

5. A CASE STUDY

A motorcycle engine design process model (Fig. 6) is used to test the simulation algorithm. The process model is built by referring to the component design structure matrix model (Fig. 7) in Tang (2009), while additional information such as the task durations and task output logics were obtained by referring to the motorcycle engine structure model and talking to three senior designers in the China Qingqi Group Company. The arrows in Figure 6 represent dependencies between design tasks, and the two lower subpanels in Figure 6 are used by process modelers to input process data related to the task and the dependency. The dependency includes component interface information and other information resources transferred between design tasks. These dependencies are not differentiated in the model, which, on one hand, makes it easy to develop graph-based change propagation algorithms, but, on the other hand, makes it difficult to obtain exact solutions for a particular design change because different causes of changes are not embodied in the model. This certainly merits further research in the future.

In the motorcycle design process, suppose an emergent design change occurs in the design task "rear crankshaft design." Figure 8 shows a set of 21 shortest paths that covers all the tasks influenced by the emergent change to the rear crankshaft design task with an ICI of 0.6. According to this figure, 25 tasks can be affected by the emergent change, which illustrates that the tasks in the process model are highly interconnected, and selection of change propagation paths is critical for finding an easy solution to satisfy the initial change requirement so that designers do not need much effort to implement the solution. Figure 9 shows the top 10 shortest change propagation paths. Because we do not constrain the tasks that each propagation path can traverse, the same design tasks can have occurrences in different change propagation paths due to their little costs for resolving design changes, which may also lead to little differences of durations between different change propagation paths for the same ICI. However, the differences between propagation paths with different ICIs are higher than the differences of paths with the same ICI, and the greater the differences of ICIs are, the higher the differences of durations are. This conclusion can be verified by Figure 10, which shows the distribution of process durations of the shortest change propagation paths with respect to different ICIs. Therefore, the selection of a proper change propagation path is critical to the product development project, especially when great design changes occur. For high ICIs, however, it seems that more and shorter change propagation paths may be found to satisfy the change requirement as the boxplots show in Figure 10.

Based on the change propagation paths shown in Figure 9, alternative solutions are available for designers to select in order to propagate a change to a certain item and not to other influenced items in order to constrain the change propagation scope. For example, in the motorcycle engine development process, a change requirement, namely, increasing 5% of the press-in driving torque of rear crankshaft, is put up forward to reduce the slippage caused by a fairly loose interference fit between the crankshaft pin and rear crankshaft. Slightly increasing the torque may greatly affect the fatigue

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

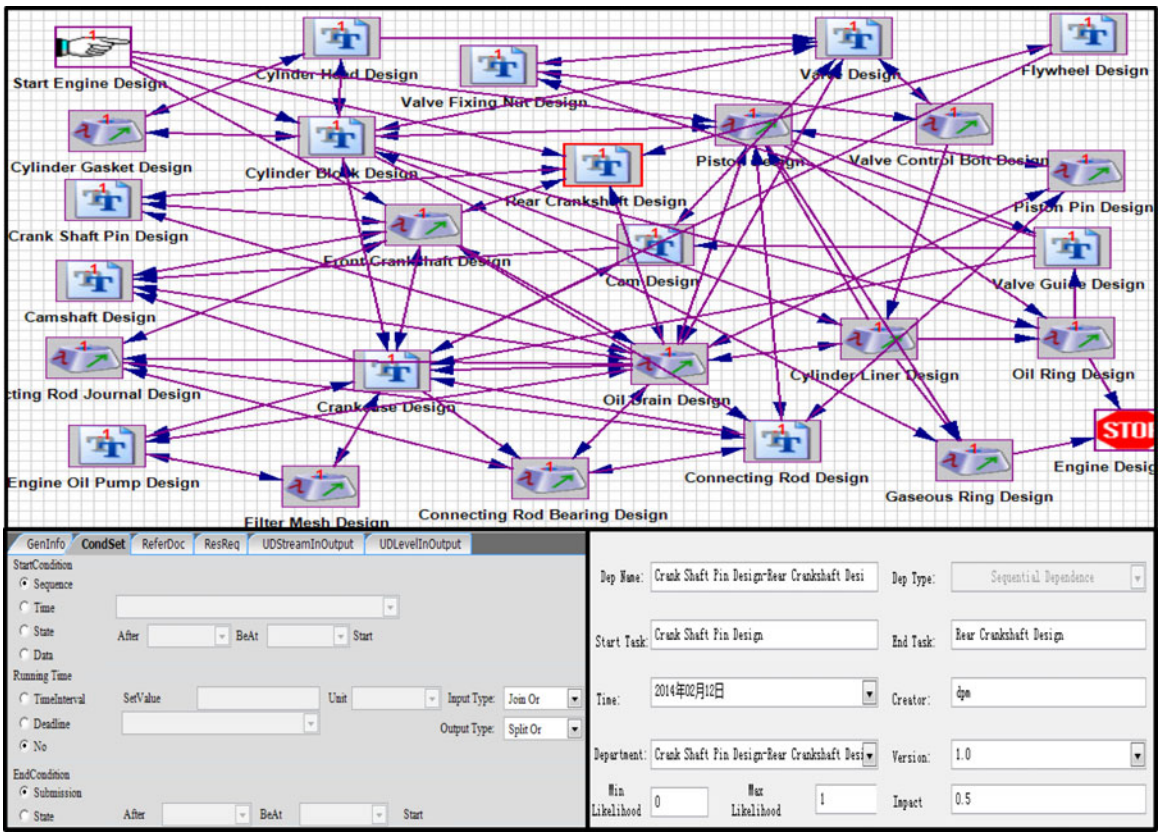

Fig. 6. A motorcycle engine design process model.

limit and balance of the rear crankshaft, and the performance of the changed rear crankshaft must be tested by experiments. More important, the change may further propagate to other components of the engine, for instance, the crankcase, connecting rod and piston, and so on, due to the complicated interconnections between these components (Fig. 6). Considering the above factors involved in this change, the designer thinks that an ICI of 0.3 may be incurred to the design of the rear crankshaft if the change is implemented. The path 1–2–3–2 is finally selected because it contains the least number of different design tasks, and in the solution, the size of the oil drain, the diameters of the rear crankshaft, and the crankshaft pin are modified. In another type of motorcycle engine development process, the authors also find a change requirement saying 8% of the total mass of the rear crankshaft needs to be reduced to improve running smoothness of the engine. This change request can generate a 0.4 change impact to the design of the rear crankshaft as evaluated by the designer. Because the rear crankshaft needs more changes, although the path 1–2–3–2 is still a solution (as shown in Fig. 9), it may not be a satisfactory one for this change because the change results from the redesign of the rear crankshaft affect not only the crankshaft pin and the oil drain but also the connecting rod journal and bearing, so another shortest path, namely, 1–2–3–6–12, is chosen, although this path has a higher sensitivity index than the former one.

It can be seen from Figures 8 and 9 that most change paths contain 4–5 tasks, while few of them only contain 3 or 6 tasks. Figure 11 shows the number of the shortest change propagation paths with ICIs from 0.1 to 0.6. If all the change propagation paths (Fig. 11) are taken into consideration, it can be generally concluded that the more tasks the change paths contain, the longer the duration for resolving the changes is, as shown in Figures 12 and 13. According to the number of design tasks and changes shown in these two figures, some design tasks may be iterated for 2–3 times in the change paths, and the higher the ICI is, the more changes to other design tasks it may create in order to completely absorb the change impact, and thus the more frequently design tasks may be iterated due to the interdependent relationships among tasks. To constrain the change propagation range, it is expected by designers that the fewer the number of design tasks and changes the paths contain, the better they are. However, with the extent to which the amount of redesign works becoming higher, more solutions (Fig. 11) may be found to resolve the initial change requirement, although more efforts may be spent on implementing solutions (Fig. 12). For the great design changes, most of the paths may be discarded by designers, considering changes may affect the great range of the whole process. However, even after these paths are removed, the remaining paths output by the algorithm provide a large number of choices for designers to select. Therefore, the

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

No.	Design Task	Output Type	Duration(Day)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	Valve Design	Split Or	2											0.2							0.7		0.5		0.8			0.3
2	Cylinder Head Design	Split Or	3	0.2			0.5														0.7							
3	Crankshaft Pin Design	Split Or	1								0.5											0.6	0.2					
4	Cylinder Gasket Design	Split Or	1.5		0.7																0.3							
5	Crankcase Design	Split Or	10												0.4	0.7	0.3		0.5	0.5		0.8	0.2					0.6
6	Piston Design	Split Or	5							0.8		0.2	0.6			0.5					0.3		0.7				0.1	
7	Gaseous Ring Design	——	2																									
8	Rear Crankshaft Design	Split Or	8			0.4																0.3	0.2	0.5				
9	Oil Ring Design	——	4																								0.3	
10	Piston Pin Design	Split Or	3						0.2							0.9							0.1					
11	Valve Control Bolt Design	Split Or	2	0.2																					0.2	0.5		
12	Filter Mesh Design	Split Or	2					0.8												0.6								
13	Connecting Rod Design	Split Or	7					0.6	0.6				0.4					0.6	0.7			0.8						
14	Camshaft Design	Split Or	6																			0.4	0.5					
15	Connecting Rod Journal Design	Split Or	4													0.8			0.4			0.8	0.6					
16	Connecting Rod Bearing Design	Split And	4													0.4		0.6					0.2					
17	Engine Oil Pump Design	Split And	9					0.7							0.5								0.4					
18	Cylinder Block Design	Split Or	10	0.7	0.9		0.2	0.7	0.3	0.8		0.2														0.6		
19	Front Crankshaft Design	Split Or	8			0.3		0.5			0.8					0.8	0.3	0.6					0.4					
20	Oil Drain Design	Split Or	6	0.3		0.2		0.4	0.4		0.3		0.3				0.6	0.4	0.7	0.8						0.7		
21	Flywheel Design	Split Or	5					0.2			0.2																	
22	Valve Fixing Nut Design	Split And	2	0.4										0.8														
23	Cylinder Liner Design	Split Or	5						0.8	0.6		0.6									0.3		0.1					
24	Valve Guide Design	Split And	7					0.4																	0.4			0.3
25	Cam Design	Split Or	8	0.6				0.8									0.8											

Fig. 7. Design structure matrix for propagation impacts between motorcycle engine component design tasks.

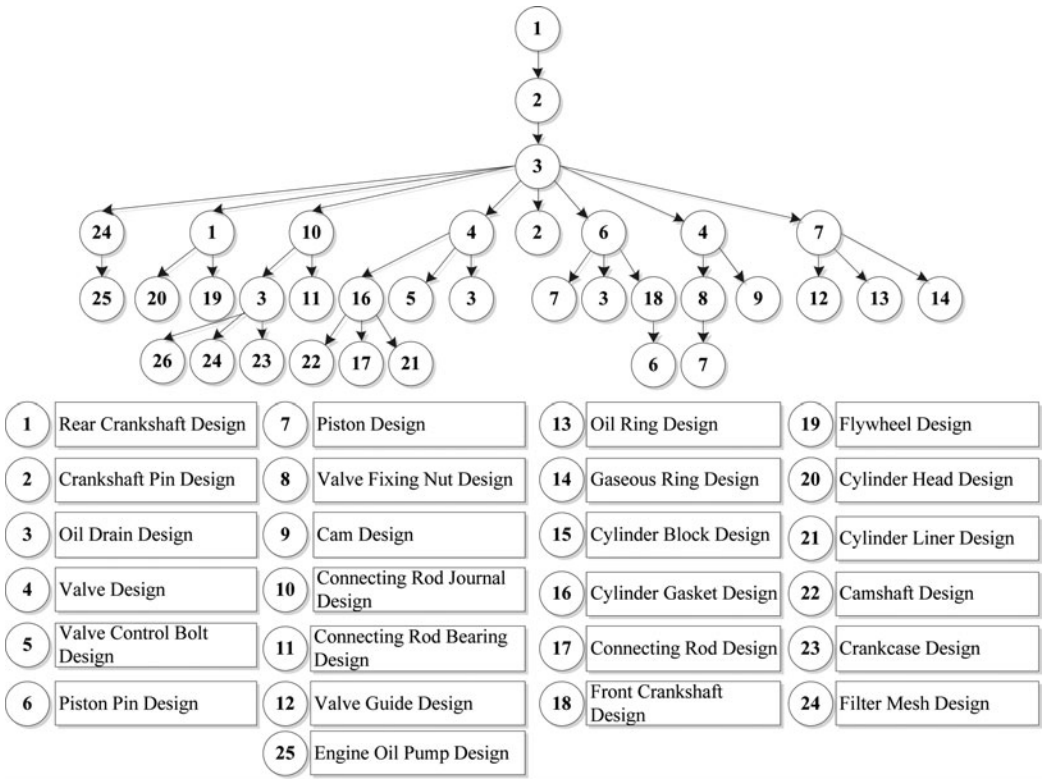

Fig. 8. A set of minimal paths that spans all the tasks influenced by the emergent change in the "rear crankshaft design" task with an ICI of 0.6.

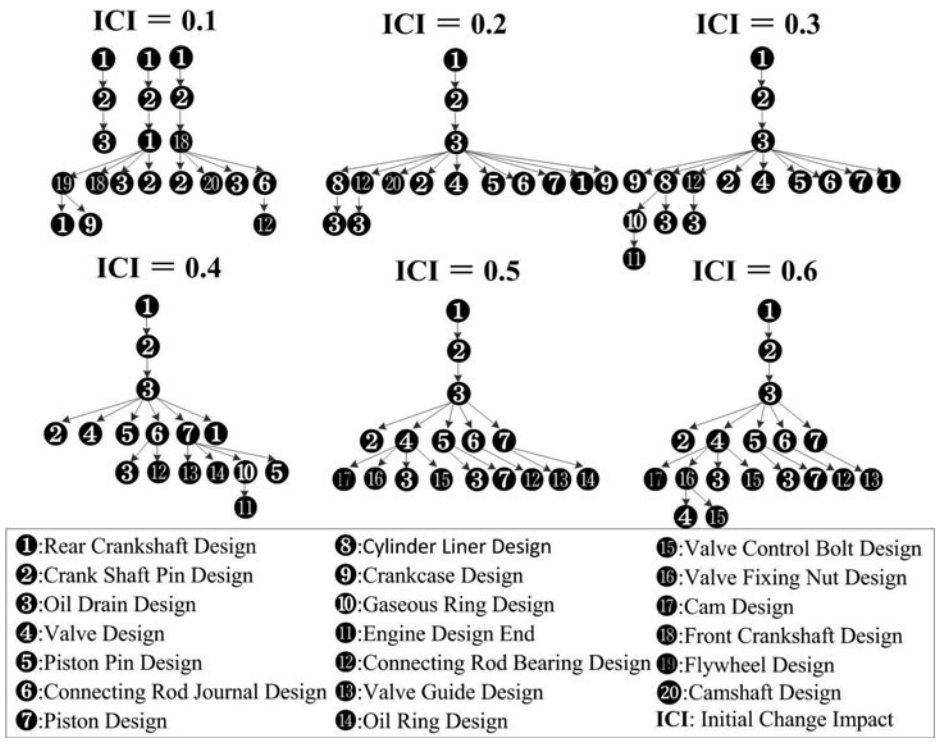

Fig. 9. Top 10 shortest change propagation paths for different initial change impacts.

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

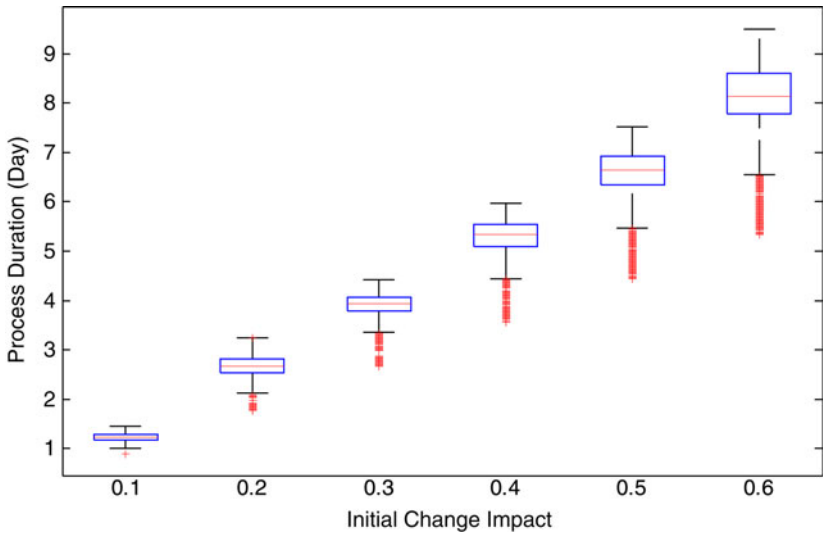

Fig. 10. Distributions of process durations of the shortest change propagation paths with respect to different initial change impacts.

algorithm can not only predict the change impacts but also provide solution choices for designers.

As for the change sensitivity of process duration with respect to the ICI, the sensitivity values for the shortest paths starting from the rear crankshaft design task are shown in Figure 14. It can be concluded from the figure that the largest portion of shortest paths in each subgroup plays the center-right part of distribution graph, which means more solutions may be found in this area. For the same ICI, the sensitivity of the long process duration is greater than the short process duration, which means the longer it takes to resolve a change request, the more sensitive the solution is to the ICI. For this reason, it is right for designers to take simple and easy-to-implement solutions to satisfy the initial change request. The sensitivities of process durations with respect to big ICIs are almost the same as the sensitivities of process durations with respect to small ICIs, which means the change to the rear crankshaft design task has a steady impact on the whole design process. In this case study, task durations are relatively short, and the duration differences between design tasks are small, which may also lead to small differences in the durations and sensitivities of the shortest paths. Certainly much more different change propagation paths and propagation sensitivities can be found if much larger differences of task durations in a process network are given.

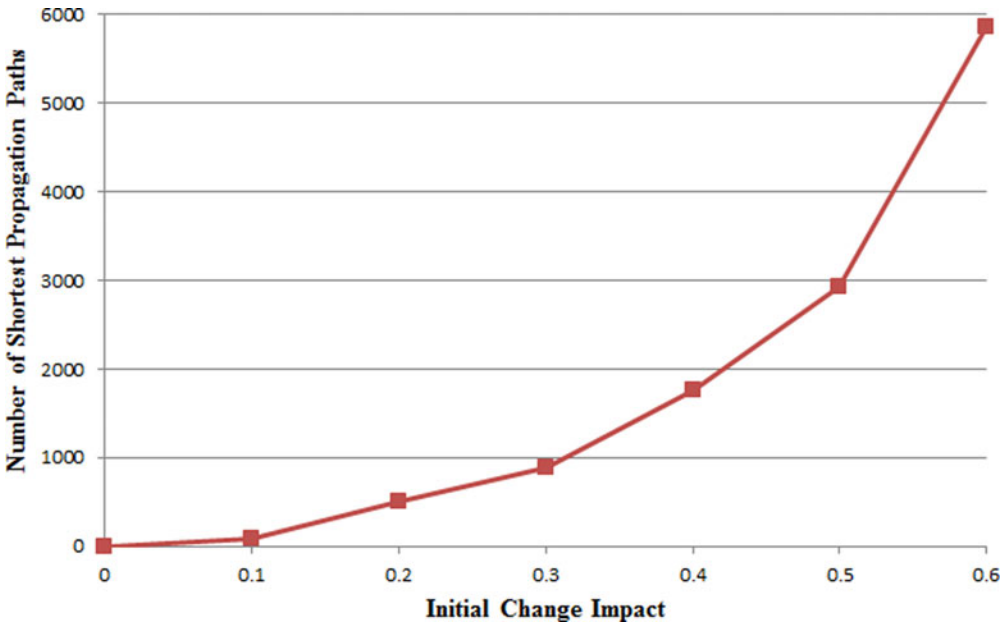

Fig. 11. Number of shortest propagation paths versus initial change impacts.

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

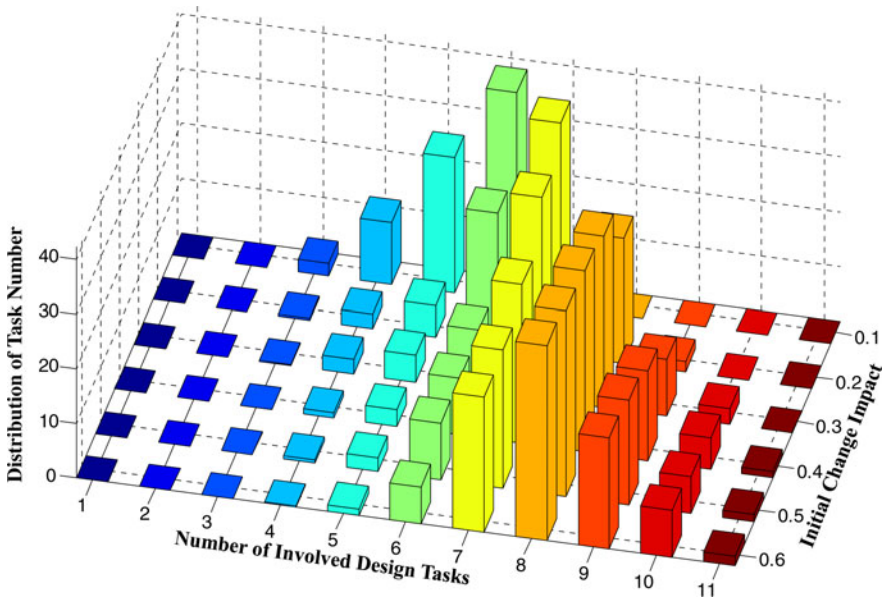

Fig. 12. Distribution of involved task numbers in the paths with different initial change impacts.

6. CONCLUSIONS AND FUTURE WORK

This paper introduces a method for managing design changes in complex engineering design processes. The innovation of the paper includes the following: the shortest path method is proposed to find the most timesaving propagation routes for sequential changes; and a local sensitivity analysis method is given to compute the design impacts that an instigating change can create on the downstream dependent design tasks. The results show that the number of change propagation paths grows exponentially with respect to the ICI, and the longest change resolution method has the highest change impact sensitivity, which proves the shortest change resolution path is the most timesaving propagation route.

Future work would be directed toward developing a pruning method to remove the shortest paths that represent unsatisfactory solutions. According to our interview with the motorcycle engine designers, although the solution for a

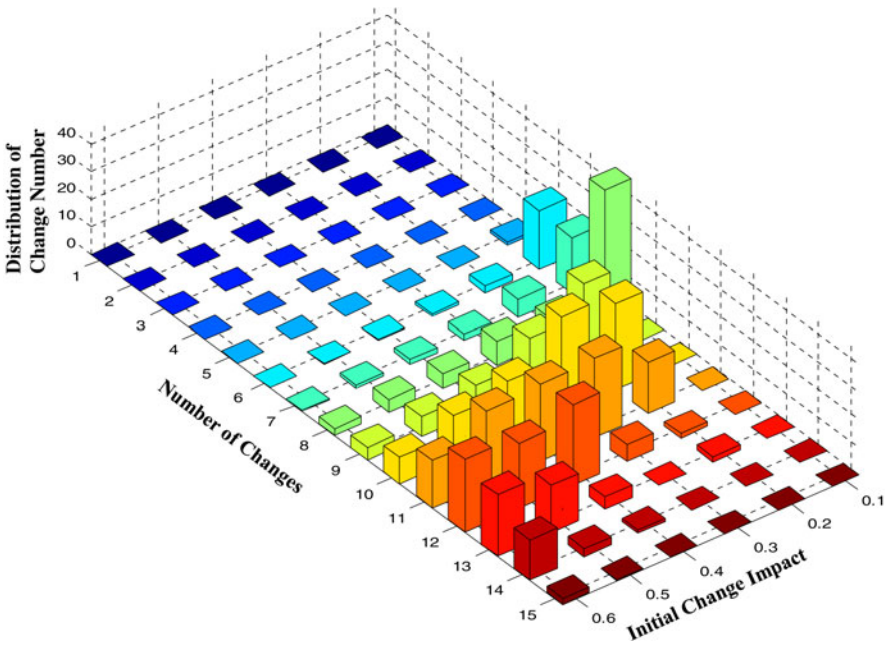

Fig. 13. Distribution of change numbers in the paths with different initial change impacts.

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

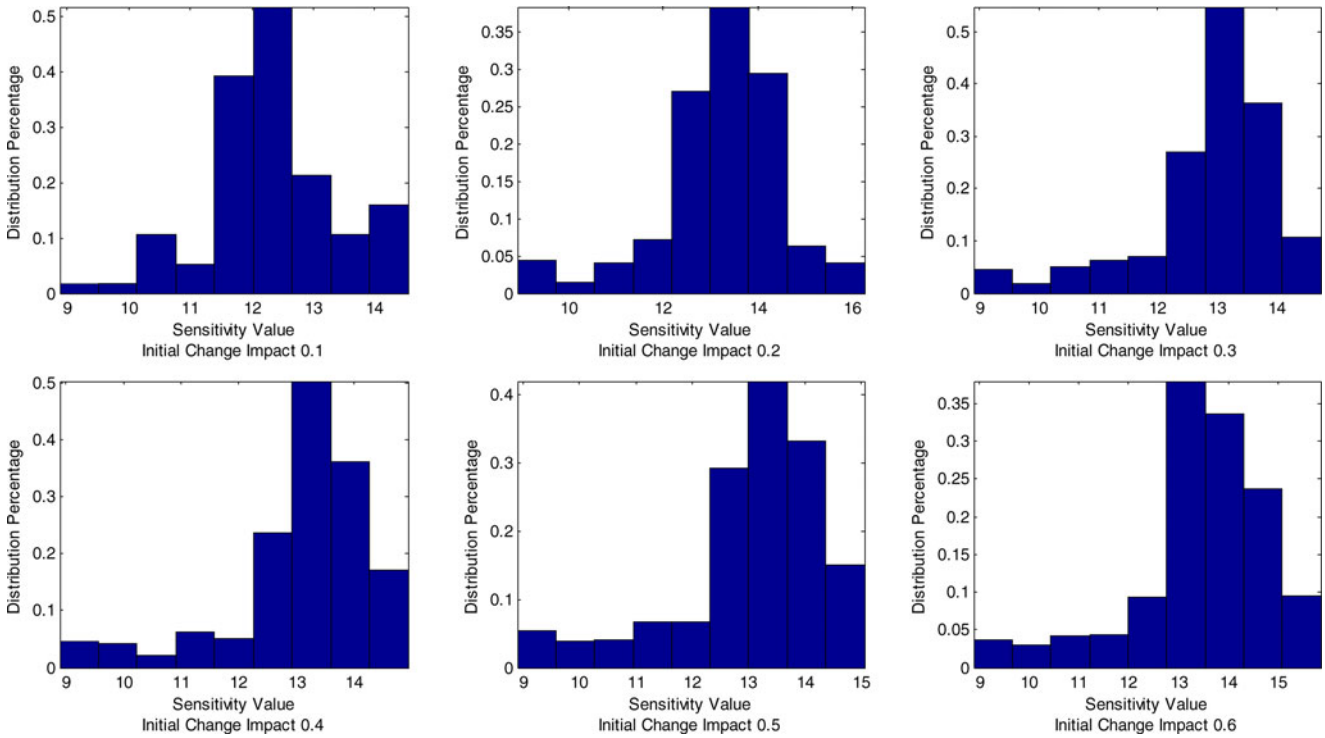

Fig. 14. Distributions of sensitivities of the shortest change propagation paths with respect to different initial change impacts.

design change can be found in the resultant shortest paths, a number of useless paths are also listed in the results. Therefore, more design constraints must be introduced into the process model to diversify the shortest paths in order to find more different solutions as well as to remove unsatisfactory ones. The method is only tested in the structure design of motorcycle engines. More design knowledge and data about the engine design process should be collected, and the method needs to be further tested in the models that span more stages of the product development process and have much larger differences of task durations in order to find paths with more different lengths for resolving a design change.

ACKNOWLEDGMENTS

This research work is mainly funded by the National Natural Science Foundation of China (Grants 51475416 and 51175457). The first author is also supported by the Fundamental Research Funds for the Central Universities of China. The third author personally acknowledges the Discovery grant supported by NSERC, Canada.

REFERENCES

Bhise, V. (2014). *Designing Complex Products with Systems Engineering Processes and Techniques*. London: CRC Press.

Bowman, R. (1995). Efficient estimation of arc criticalities in stochastic activity networks. *Management Science 41(1)*, 58–67.

Browning, T., & Eppinger, S. (2002). Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management 49(4)*, 428–442.

Clarkson, J., Simons, C., & Eckert, C. (2004). Predicting change propagation in complex design. *Transactions of the ASME Journal of Mechanical Design 126(5)*, 788–797.

Dijkstra, V. (1959). A note on two problems in connection with graphs. *Numerische Mathematik 1*, 269–271.

Dodin, B., & Elmaghraby, A. (1985). Approximating the criticality of the activities in pert networks. *Management Science 31(2)*, 207–223.

Eckert, C., Clarkson, J., & Zanker, W. (2004). Change and customization in complex engineering domains. *Research in Engineering Design 15(1)*, 1–21.

Eppinger, S., Whitney, D., Smith, R., & Gebala, D. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design 6(1)*, 1–13.

Giffin, M., de weck, O., Bounova, G., Keller, R., Eckert, C., & Clarkson, J. (2009). Change propagation analysis in complex technical systems. *Transactions of ASME Journal of Mechanical Design 131(8)*, 081001.

Hamraz, B., Caldwell, N., & Clarkson, J. (2013*a*). A matrix-calculation-based algorithm for numerical change propagation analysis. *IEEE Transactions on Engineering Management 60(1)*, 186–198.

Hamraz, B., Caldwell, N., & Clarkson, J. (2013*b*). A holistic categorization framework for literature on engineering change management. *Systems Engineering 16(4)*, 473–505.

Jarratt, T., Eckert, C., Caldwell, N., & Clarkson, P. (2011). Engineering change: an overview and perspective on the literature. *Research in Engineering Design 22(2)*, 103–124.

Kerzner, H. (2009). *Project Management: A Systems Approach to Planning, Scheduling and Controlling*. Hoboken, NJ: Wiley.

Krishnan, V. (1996). Managing the simultaneous execution of coupled phases in concurrent product development. *IEEE Transactions on Engineering Management 43(2)*, 210–217.

Lee, H., Seol, H., Sung, N., Hong, Y., & Park, Y. (2010). An analytic network process approach to measuring design change impacts in modular products. *Journal of Engineering Design 21(1)*, 75–91.

Li, Y., Zhao, W., & Shao, X. (2012). A process simulation based method for scheduling product design change propagation. *Advanced Engineering Informatics 26(3)*, 529–538.

Mummolo, G. (1997). Measuring uncertainty and criticality in network planning by PERT-path technique. *International Journal of Project Management 15(6)*, 377–387.

https://doi.org/10.1017/S0890060415000311 Published online by Cambridge University Press

Pahl, G., Beitz, W., Feldhusen, J., & Grote, K. H. (2007). *Engineering Design: A Systematic Approach*, 3rd ed. London: Springer.
Siddiqi, A., Nounova, G., de weck, O., Keller, R., & Robinson, B. (2011). A posteriori design change analysis for complex engineering projects. *Transactions of ASME Journal of Mechanical Design 133(10)*, 101005.
Tang, D. (2009). *DSM Based Product Design and Development*. Beijing: Science Press (in Chinese).
Viteri, J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory 13(2)*, 260–269.
Wilson, J. (1996). *Introduction to Graph Theory*, 4th ed. Essex: Addison–Wesley.
Wynn, D.C., Caldwell, N., & Clarkson, J. (2010). Can change prediction help prioritise redesign work in future engineering systems? *Proc. Int. Design Conf.—DESIGN 2010*, Croatia, May 17–20.
Yang, F., & Duan, G. (2012). Developing a parameter linkage-based method for searching change propagation paths. *Research in Engineering Design 23(4)*, 353–372.

Yuliang Li is an Associate Professor in the Department of Mechanical Engineering, Zhejiang University. He attained his PhD and BS degrees from Huazhong University of Science & Technology and Shandong University of Technology, respectively. His current research interests include design evolution, product design process modeling, and product conceptual optimization.

Wei Zhao is a Lecturer at Zhejiang University of Finance and Economics.

Y.-S. Ma is a Professor in the Department of Mechanical Engineering, University of Alberta, and a registered Professional Engineer with APEGA, Canada. He received his BEng degree from Tsinghua University, Beijing, and MS and PhD degrees from the University of Manchester Institute of Science and Technology. Dr. Ma served as an Associate Editor of *IEEE Transactions of Automation Science and Engineering* from 2009 to 2013 and has been an editor of *Advanced Engineering Informatics* since 2012. He won the ASTech award sponsored by the Alberta Science and Technology Leadership Foundation jointly with Drader Custom Manufacturing Ptd. in 2012. He currently teaches capstone design projects, engineering economics, and manufacturing processes. His research areas include interdisciplinary heavy oil recovery production tooling engineering, feature-based product and process modeling, plastic molding simulation and mold design optimization, CAD/CAE integration, CAD/CAM, ERP informatics modeling, and product lifecycle management.