INTERNATIONAL DESIGN CONFERENCE - DESIGN 2010
Dubrovnik - Croatia, May 17 - 20, 2010.

DESIGN 2010

CAN CHANGE PREDICTION HELP PRIORITISE REDESIGN WORK IN FUTURE ENGINEERING SYSTEMS?

D. C. Wynn, N. H. M. Caldwell and P. J. Clarkson

Keywords: engineering change prediction, process simulation

1. Introduction

In many complex engineering industries, companies are required to develop more complex products with shorter lead times and greater cost-effectiveness. The approach being adopted in aerospace companies across Europe is to integrate and automate design tools and methods to support design, simulation, testing and certification, of aircraft and their sub-systems in virtual environments. The enabling technologies currently under development are intended to provide designers with the ability to more readily access information and more quickly perform design tasks. Consequently, the dynamic complexity of the design process is expected to increase, driven by faster iteration cycles and a denser network of dependencies among the actors performing different tasks. Dynamic complexity will be further increased by easier communication and therefore easier design iterations across intra- and inter-organisational boundaries, all made possible by virtual computing environments and the distributed enterprise.

As a result of easier and faster iterations during the design process, the management of design changes, their propagation and their impacts are likely to become increasingly important. This paper investigates whether, and to what degree, a better understanding of how change propagates could help mitigate some of the risks to project performance that will arise from the increased dynamic complexity in future design environments.

The paper is organised as follows. In Section 2, we provide background and motivation by discussing how many companies expect design will be conducted in future, highlighting the increasing importance of managing design change. We review existing change management tools and consider how they can support this new challenge. Section 3 states the specific research questions addressed by this paper and explains the simulation-based approach we adopted to explore them. Section 4 describes our simulation model in depth. Section 5 uses a case study of helicopter design as the basis for experimentation to explore the research questions. Section 6 presents our findings and highlights their implications for change management research. Section 7 discusses the validity and limitations of our experimental approach and highlights several opportunities for further work. Section 8 concludes the paper.

2. Background

2.1 The increasing importance of engineering change management

Many engineering companies and design software vendors today are involved in the development of next-generation design and optimisation systems, intended to replace physical modelling, testing and potentially certification within a company or department with simulations conducted on "virtual

products". These future design systems aim to enable seamless information transfer between tools and sites, enabling designers to access and change information much more easily and thereby increasing interdependencies in the process. In turn, this should create faster design cycles through increased concurrent engineering.
Any one designer within a complex process interacts with the decisions of others through consideration and modification of related design information. Each design iteration therefore leads to changes in the assumptions, models and analyses of other process participants, potentially requiring knock-on rework. As integrated systems will enable easier access to data, there is potential to initiate these "bow waves" of rework more frequently. Changes may also be more likely to propagate than in today's systems, where designers are often forced to contain iterations within individual tools and sub-problems due to the overheads of model acquisition and conversion. As a result, work is often based on a given set of input information and only released once a relatively final version is achieved. Easier access to information and faster computation could lead to smaller and more frequent updates, which in turn could cause change to propagate further and more frequently.
In summary, although future design systems will enable design tasks and workflows to be completed more quickly, they also have potential to increase change and its propagation due to increased dynamic complexity in the overall design process. The overhead of handling this additional rework has potential to slow the progress of projects, and therefore there will be a greater need to manage change effectively. One way to support the coordination of change activity is through a better understanding of how changes propagate.

2.2 Research in change propagation and prediction

The benefits of concurrent engineering in terms of time and cost savings have been clear for some time, as has the importance of effective coordination and collaboration to ensure the timely flow of information generated during product development [O'Brien and Smith 1995]. This section reviews research on engineering change propagation and how it relates to information coordination, prior to discussing tools which have been developed to support change prediction.

2.2.1 The nature of engineering change propagation

There have been a number of empirical studies to ascertain the nature and the importance of change in the design and development process. Eckert et al. [Eckert et al. 2004] undertook interviews at Westland Helicopters Limited to investigate the nature of change, distinguishing between emergent change wherein a problem with the design required change to resolve it and initiated change arising from new requirements. They further identified that product elements can absorb, propagate or multiply changes as they propagate via component and subsystem interactions, and highlighted different strategies for handling change in terms of forward partial redesign and backwards patching/debugging. Other studies have analysed historical case data to classify the sources of change stimuli and the areas of a design most impacted by change. In a case study on a temporally contiguous set of engineering changes in the detailed design phase of aircraft carriers, Rowell et al. [Rowell et al, 2009] reported that just over two-thirds of the cases could not be specifically classified in terms of originating stimuli, but half of the effects caused by the changes affected one or other of bill of materials, the baseline or the structure. Giffin et al. [Giffin et al. 2009] reported statistical evidence of the existence of patterns of change propagation effects and parent, child and sibling relationships among systems affected by changes, based on a case study involving 41,500 change requests raised over an eight-year design period of a large sensor system.
Drawing on a series of studies involving thirteen companies across seven industries, Fricke et al. [2000] proposed five strategies to better manage change. These were: change prevention; front-loading of change; better assessment of the necessity and benefits of specific changes; efficient implementation of changes in terms of cost, time and resources; and finally learning how to improve efficiency and effectiveness of future changes from lessons learned from previous changes [Fricke et al. 2000]. Rouibah and Caskey likewise consider the difficulties in managing engineering change of products across organisations, indicating that communication, collaboration, and working towards consensus are key to successful change management [Rouibah and Caskey 2003].

2.2.2 Change prediction tools and techniques

A number of tools and approaches have been proposed to help manage engineering change by predicting its impact. These tools are based on the concept that change propagates between parameters or components through the linkages between them. Many are based on Dependency Structure Matrices (DSMs, e.g., Figure 1) which represent the elements of the product and their interrelationships in a compact and accessible form. DSMs have been employed by various authors as the basis for modelling change propagation, by augmenting the basic matrices with additional information such as the likelihood of a change propagating between two connected elements and the impact on the dependent element if the change occurs. Based on this representation, various algorithms to predict indirect change propagation have been developed, such as the CPM technique [Clarkson et al. 2004]. Zhao et al. use a similar activity-based DSM technique to model the information flows in a case study of a construction project, and apply a Monte Carlo simulation algorithm to investigate the potential rework of activities and how they vary in terms of mean duration, etc., as a result of changes [Zhao et al, 2008]. The Change Propagation Analysis approach [Rutka et al. 2006] uses a dependency model of heterogeneous information, adding factors such as multiple types of change at both the initiating item and affected items, and generating global risk values by following change propagation paths. Software implementations of these change prediction methods have mostly been limited to research prototypes and relatively small models.

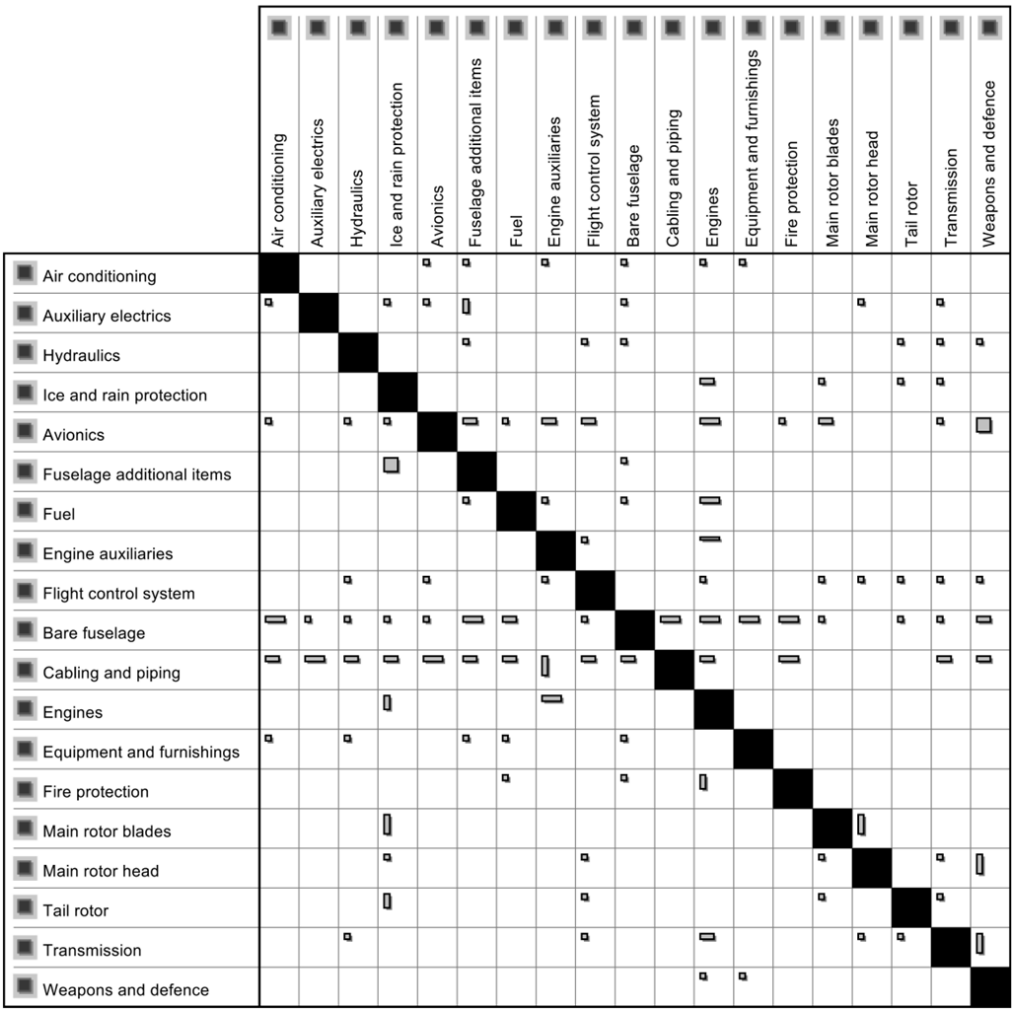


Figure 1. Westland Helicopters EH101 CPM model, showing subsystems and the connections between them [Clarkson et al. 2004]

2.2.3 Opportunities to apply change prediction

To recap, we have argued that the technology being developed to support design in industries such as aerospace will involve an increase in concurrency and the potential for faster design iterations by increasing the ease of accessing, modifying and transferring design data. In turn, this has potential to create more engineering change and to increase the importance of managing it effectively.

One means of managing change is to reduce change initiation, for example by freezing decisions. However, many decisions cannot be truly frozen until very late in the design process. Additionally, since iteration is a fundamental part of the design process, required to account for new information generated during design, it cannot be avoided entirely. Another approach is to more effectively manage the rework caused by change after it has been initiated. In this paper, we investigate how the effects of change can be dealt with in an appropriate order to minimise unnecessary knock-on rework, through dynamic scheduling of the affected workflows. In particular we consider how change prediction can assist this scheduling – exploring whether, and to what degree a better understanding of change propagation can improve change process efficiency.

3. Research questions and methodology

In the remainder of this paper we thus explore the following research questions:

1. Does change prediction help to make rework prioritisation decisions which result in better process performance than decisions made without awareness of propagation effects?
2. How much effort should be put into change prediction? i.e., is greater effort invested in more accurate prediction reflected in better process performance?

We explore these questions though simulation experiments, as real-world complex design processes are difficult to access, instrument or modify in practice. We develop a new simulation approach and employ it to look at the impact of sequencing change operations upon the total duration of a process. We base our simulations on a real-world model of change propagation: the Westland Helicopters EH101 change prediction model earlier reported by Clarkson et al. (2004).

4. Simulation model

The computational model we developed to explore the questions stated above simulates the process of responding to change requests and completing all knock-on rework. The simulation proceeds in four stages, which are shown in overview in Figure 2 and described in detail below.

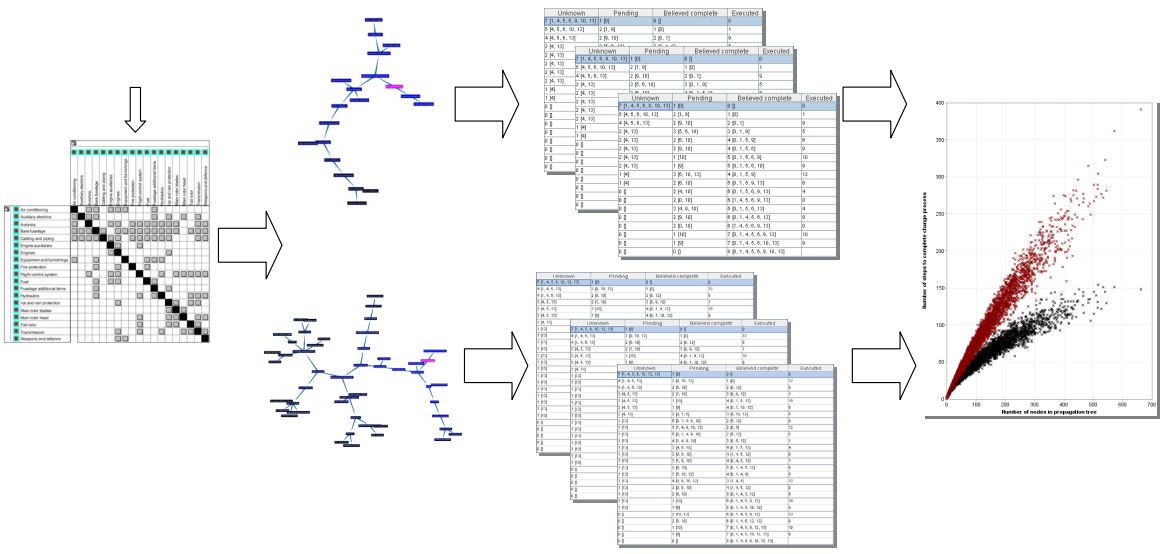


Figure 2. Overview of the simulation algorithm

4.1 Stage 1 - Model the product and select initiating component to investigate

The simulation was developed to operate on models of change propagation as constructed for the CPM approach [Clarkson et al. 2004], without requiring additional data. This allows investigation of change process behaviour based on the datasets already available for that tool. The model we used for our simulation experiments is shown in Figure 1. Each dependency shown in this figure is supplemented by the likelihood that a change initiated in the subsystem named in the column header will propagate to that named in the row header. These likelihood and impact values are indicated by the width and height of the grey rectangles representing each dependency. We use this model to simulate the process of responding to a change made in a given initiating component.

4.2 Stage 2 - Sample propagation trees

We assume that change propagation follows dependency linkages between elements, and that the likelihood of propagation from any one element to each connected element can be estimated as in the CPM model [Clarkson et al. 2004]. Thus, change propagates outward from the initiating component and forms a propagation tree. According to the likelihood/impact model of change propagation, the depth of this tree is not limited. However, prediction models such as the CPM assume in addition that each component can only be changed once. We remove this assumption to allow cycles of change activity – in other words, such that a change to component A can result in several knock-on changes, which may ultimately require A to be changed again. This behaviour is common in practice, as noted by Fricke et al. in their analysis of case studies, where they identified instances of "closed loop" changes where a chain of changes led in several steps back to the initial solution [Fricke et al. 2000]. Similar findings are reported by Giffin et al. (2008).

In our approach, a Monte-Carlo method is used to sample from the space of possible change trees, including the possibility for cycles. A given cell *ij* in the CPM matrix indicates that change in a given component *i* propagates to cause change in another component *j* with probability l_{ij}. When the propagation from component *i* is simulated, propagation to component *j* occurs only if a uniformly-distributed random number in the range 0 to 1 is less than l_{ij}. If this occurs, a new node is added to the propagation tree and the procedure is repeated considering *j* as the next component from which change can propagate further. Since the sum of outward propagation likelihoods for some components in Figure 1 is greater than 1, the propagation tree can become very broad and as a result the Monte-Carlo sampling algorithm does not always terminate in reasonable time. To prevent this, a maximum propagation depth of 10 steps is used in our experiments. We believe this is a reasonable assumption, as in practice changes will not be allowed to propagate indefinitely. While limiting the maximum depth does bias the shape of larger trees, which will have a larger breadth/depth ratio due to the truncation, in the resulting datasets described in Section 6 the general trends are clear.

Each sample which is generated represents a different change propagation tree which could occur following an initiated change to the specified component. The actual propagation path which would occur in practice depends upon the detail of the specific change and how it is managed, for instance including recovery actions taken by designers to limit the extent of change propagation.

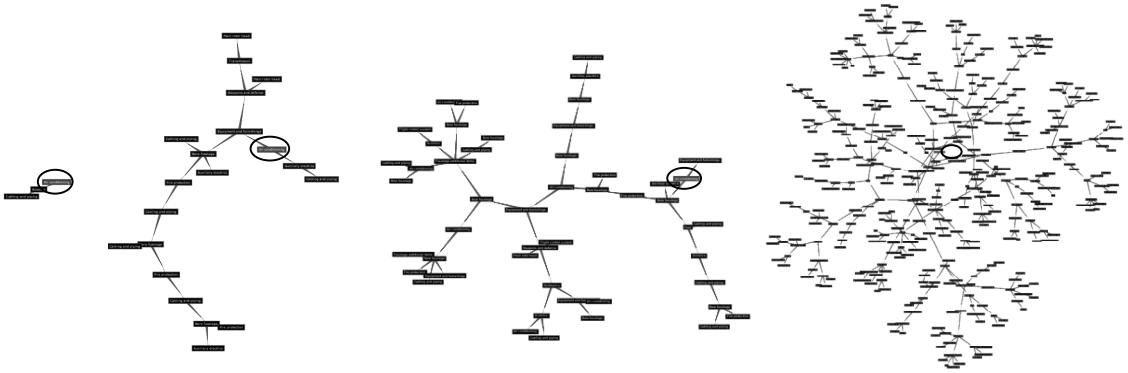


Figure 3. Sampled EH101 propagation trees with truncation depth 10, comprising 3, 20, 50, 400 nodes respectively. Change ripples outwards from the circled nodes.

4.3 Stage 3 - Simulate implementation processes

Each propagation tree generated in Stage 2 (Section 4.2) is used as the basis of simulating the possible implementation processes for that change scenario.

4.3.1 Overview of simulation approach

The process simulation is based on the assumption that the rework arising from each change tree is scheduled and implemented by project personnel who cannot foresee the entire propagation tree before the work is implemented. In other words, as a knock-on change is implemented, downstream rework is discovered and must be dealt with in turn. This limited foresight reflects the reality that most engineers specialise in their discipline, possessing less expertise in related areas, and that no single engineer has a complete detailed overview of a large complex system and especially how change might propagate through it [Eckert et al. 2009].

Our simulation model further assumes that change implementers are faced with a choice when multiple components are observed to require change – namely, which of the component changes should be attempted next? We simulate this choice, assuming that engineers make rational decisions based on the information available to them at the time when those decisions are made.

The performance penalty assessed by our model is that if a poor decision is made, unnecessary rework is performed to change a component which will subsequently have to be changed again (although some rework is inevitable in a large propagation tree, since components can occur many times in one path due to cycles of change activity). When unnecessary rework happens, more steps are required to complete the entire change process, which hence takes longer and consumes more resources. We do not model the time taken to complete a change request, but only the number of steps. In real terms, while the number of steps is not a true indicator of implementation cost, it is a significant contributing factor which represents the overheads involved in switching tasks, such as communication, obtaining up-to-date information, etc.

4.3.2 Representation of process state

The algorithm is based on step-wise transitions between discrete process states. The state of the process is modelled as three sets of *rework tasks*:

1. *Unknown* - rework tasks for components which require change, but are not known to the simulated designer yet;
2. *Pending* - rework tasks for components for which changes are known to exist by the simulated designer;
3. *Believed* complete - rework tasks for components where rework associated with this change tree has already been executed at least once in the simulated process.

A single rework task exists for each unique component in the propagation tree. Each rework task is a member of exactly one of the three sets at all times during simulation. Each rework task comprises a component and the items of rework, if any, which are currently required on that component. Items of rework are represented as the node(s) in the propagation tree which represent the rework required at the current time. The propagation tree nodes associated with a rework task determine the knock-on rework (i.e., items of rework to be added to the rework tasks for other components) which will be revealed when that rework task is completed.

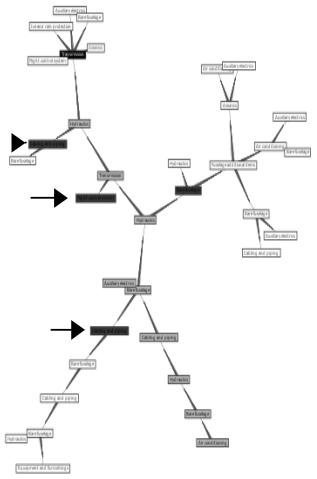
4.3.3 Algorithm

The simulation algorithm then operates as follows:

1. **Initialisation.** Unknown is populated with a rework task for every unique component in the propagation tree resulting from Stage 2 (Section 4.2). The rework task representing the single initiating component is then moved from Unknown into Pending, and it is associated with the root node of the propagation tree. This reflects the designers' knowledge that the initiating component requires change at the outset, alongside the information (which the designer may not know) regarding which items of rework will be created when that component is changed.

2. **Task selection.** A prioritisation policy is used to identify which of the rework tasks in Pending should be attempted next. The model assumes that only one such task can be undertaken at a time. Prioritisation policies are discussed further in Section 5.
3. **Task execution**. The rework task selected in Step 2 is executed. To simulate its completion, the nodes in the propagation tree representing the items of rework associated with that task at the current time are consulted. If one or more of those nodes has children, this indicates that completion of the rework task creates knock-on changes – one for each child node of each item of rework associated with the task. If knock-on change is created, the components requiring knock-on change are first identified from these nodes in the propagation tree. The rework task for each such component is moved from Unknown (if the component has not yet been visited) or Believed complete (if it has been visited at least once) into Pending. If a rework task requiring knock-on change is already in Pending, it is not moved. All tasks for which additional items of rework were revealed are then supplemented by adding the propagation tree nodes which represent those items of rework to the respective tasks. Since no rework is now pending on the task which was just completed, all propagation tree nodes associated with that rework task are disassociated from it. The newly-completed task is finally moved from Pending into Believed complete.
4. **Steps 2 and 3 are** repeated until Believed complete **contains all rework tasks**.

As the simulated process progresses, therefore, rework is progressively discovered by the simulated designer and components move from *Unknown* into *Pending*, then back and forth between *Pending* and *Believed complete* until every node in the propagation tree has been visited and all rework items are completed.



Unknown	Pending	Believed complete
10 [17, 1, 2, 3, 4, 5, 8, 9, 10, 12]	1 [0]	0 []
9 [17, 1, 2, 3, 4, 5, 8, 10, 12]	1 [9]	1 [0]
8 [17, 1, 3, 4, 5, 8, 10, 12]	1 [2]	2 [0, 9]
7 [17, 1, 3, 4, 5, 8, 12]	1 [10]	3 [0, 2, 9]
7 [17, 1, 3, 4, 5, 8, 12]	1 [9]	3 [0, 2, 10]
6 [17, 3, 4, 5, 8, 12]	3 [1, 2, 10]	2 [0, 9]
5 [3, 4, 5, 8, 12]	4 [17, 1, 9, 10]	2 [0, 2]
5 [3, 4, 5, 8, 12]	3 [17, 9, 10]	3 [0, 1, 2]
4 [3, 4, 5, 12]	4 [2, 8, 9, 10]	3 [17, 0, 1]
4 [3, 4, 5, 12]	4 [17, 8, 9, 10]	3 [0, 1, 2]
2 [5, 12]	6 [1, 3, 4, 8, 9, 10]	3 [17, 0, 2]
2 [5, 12]	5 [1, 4, 8, 9, 10]	4 [17, 0, 2, 3]
2 [5, 12]	4 [4, 8, 9, 10]	5 [17, 0, 1, 2, 3]
2 [5, 12]	3 [4, 9, 10]	6 [17, 0, 1, 2, 3, 8]
2 [5, 12]	2 [9, 10]	7 [17, 0, 1, 2, 3, 4, 8]
1 [12]	3 [2, 5, 10]	7 [17, 0, 1, 3, 4, 8, 9]
1 [12]	5 [0, 2, 4, 9, 10]	5 [17, 1, 3, 5, 8]
1 [12]	5 [1, 2, 4, 9, 10]	5 [17, 0, 3, 5, 8]
1 [12]	4 [1, 4, 9, 10]	6 [17, 0, 2, 3, 5, 8]

Figure 4. Steps in the simulated change process for a given propagation tree

To illustrate one step in the algorithm, consider Figure 4. This shows a single simulated propagation tree and part of one possible process for executing the associated rework. The propagation tree of Figure 4 (left) illustrates the step in the process which is represented by the highlighted row of Figure 4 (right). The grey nodes represent items of rework associated with the three rework tasks highlighted in the *Believed complete* column of the table; the black boxes show items of rework associated with the *Pending* tasks; and the white boxes indicate items of forthcoming rework which have not yet been 'discovered' by the simulated designer. The arrows indicate three items of rework associated with the same rework task at this step in the simulation; when that task is executed, all changes one step downstream from all three items will be revealed together.

This algorithm has similarities with the rework cycle used in many stock-and-flow (i.e., system dynamics) simulation models of design processes; the rework cycle is discussed in detail by Ford and Sterman (1998). However, unlike the rework cycle, the approach presented here allows evaluation of different change prediction capabilities, as described in forthcoming sections.

4.4 Aggregate results

Each simulated change process for each simulated propagation tree is summarised as the total number of steps in that process relative to the number of nodes in the tree. The larger the number of steps, the less efficient the process due to unnecessary rework caused by poor scheduling.

5. Experiments

To recap, the objective of this paper is to explore whether, and to what degree change prediction can help to minimise unnecessary rework, i.e. to assist the designer in determining an order for attempting work which revisits each component as few times as possible and therefore minimises the total number of steps required to complete the process. In our simulation, the independent variable which can be changed to represent change prediction capability is the task prioritisation policy mentioned in Step 2 of Section 4.3.3. In the experiments described below, we investigate whether the capability to 'look ahead' in the propagation tree improves change process efficiency.

We base our analysis on a change propagation model of a helicopter, shown in Figure 1. The model was created through a series of interviews and group knowledge elicitation exercises involving designers and senior engineers at GKN Westland [Clarkson et al. 2004].

Our experiments apply the algorithm described above to this model in order to compare the impact of different prediction capabilities, keeping the same instigating component and testing against the same set of randomly generated change propagation trees. The policies compared are:

1. **Minimise change likelihood.** Each component which requires change is ranked by the total likelihood of that component receiving change, as specified in the CPM matrix. The component least likely to receive change is chosen to change next. This is based on the simple heuristic that changing the least likely component to receive change in future minimises the likelihood of change re-occurring later. We use this policy to represent designers' behaviour if a statistical understanding of how change might propagate is available, for instance through experience of similar projects or through consultation of a CPM matrix such as that shown in Figure 1.
2. **Minimise number of steps with N-step look-ahead capability.** For each component which requires change, the tree of all knock-on changes as given by the propagation tree is examined and the change process is 'simulated'. This policy represents the effect of designers not only having a statistical understanding of how change propagates, but also being able to look ahead at how change will propagate for the specific change at hand and make the best possible scheduling decision accordingly.

   The policy operates by exhaustively testing all possible sequences of component change from the current state forward to depth N. The next component to change is then selected in order to minimise the shortest number of steps required to complete the change process. If it were possible to foresee the entire propagation tree, it would be possible to make an optimal decision using this approach. However, we assume that the designer has limited overview and can only foresee N steps of change propagation. The algorithm accounts for this by assuming that at the limit of foresight (ie. at depth N from each component which is a candidate to change next) components are selected for rework in the order which minimises the likelihood of them receiving change. In other words, it is assumed that once foresight is exhausted, decisions are made according to the same statistical understanding of change propagation explained in the first policy.
3. **Maximise change likelihood.** The component most likely to receive change is chosen next. This policy is intended to reflect a sequence of very poor choices. It provides a baseline to compare the other policies.

The change process was simulated for 100 simulated propagation trees using a truncation depth of 10. The policy incorporating look-ahead capability was evaluated for N=1, 2, and 3, and compared to a purely statistical understanding of change propagation and to the 'bad choice' policy. Simulation required 8 hours on a desktop PC. Due to the expense of computing the search required for the second policy, it was not possible to examine the impact of this policy if look-ahead capability N>3.

6. Results

The simulation results are summarised in Figure 5. The chart compares the five different task selection strategies, plotting the number of nodes in the propagation tree (x-axis) against the total number of steps needed to complete the change process (y-axis). Reading down the chart, each quintuple of points in a column represents the application of the five task selection policies to the same sampled change propagation tree.

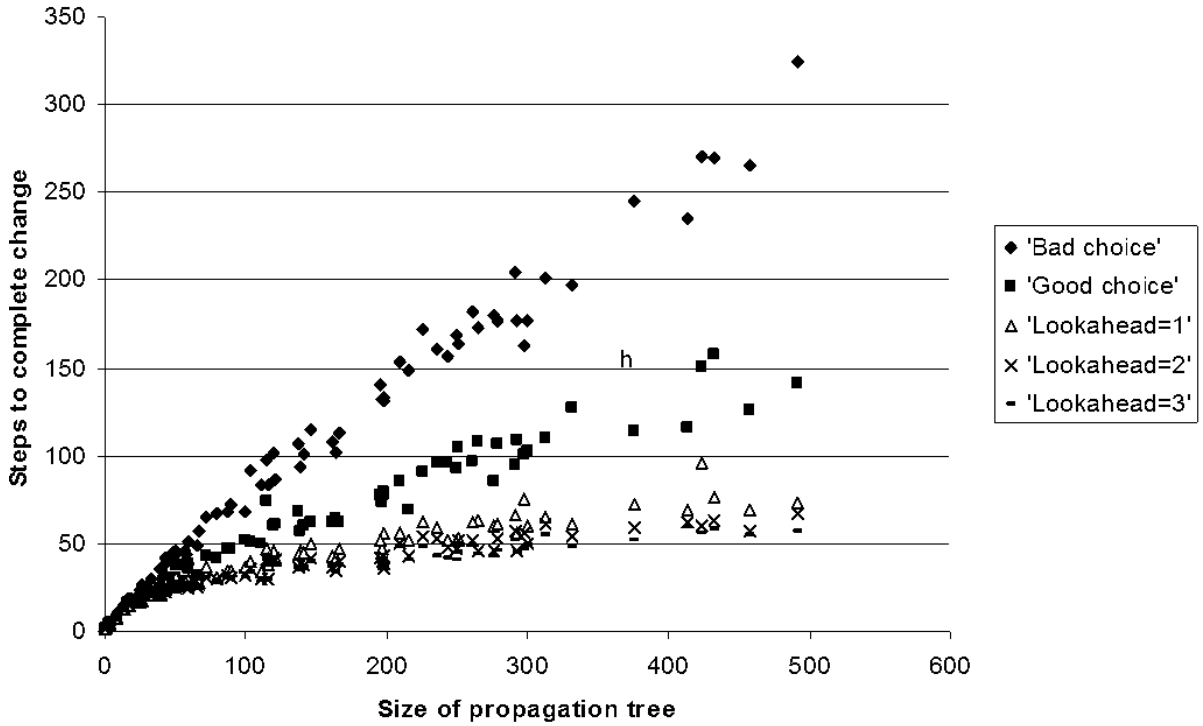


Figure 5. Effect of different change prediction capabilities on number of process steps required to implement the change

6.1 Analysis

Revisiting the research questions posed in Section 3, the following conclusions regarding the efficacy of change prediction to support change process scheduling can be drawn from this plot:

1. Poorly scheduling a change process where change can propagate in cycles can create significant unnecessary rework. Some of this rework can be avoided by understanding how change propagates and by scheduling work accordingly.
2. The scope for unnecessary rework increases with the size of the change propagation tree. Small, localised changes are difficult to mismanage from a scheduling perspective, owing to the infrequent occurrence of cycles of change causing rework to the same component. In practice, although many changes may be localised, others can involve significant change propagation. For instance, Giffin et al. identified real cases in aircraft system design where the number of nodes in a change network exceeded 2000 [Giffin et al. 2009]. Thus, in complex problems the change scheduling effects identified in this paper may be significant.
3. A statistical understanding of how change propagates – such as that provided by the change prediction tools reviewed in Section 2.2 – is helpful for scheduling and allows significant improvement over a bad decision.
4. Significant further improvements can be made by understanding how a change will propagate *in the specific case under consideration.* By being able to predict change propagation only one step ahead, it is possible to avoid substantial unnecessary rework in the implementation process which is caused by inefficient organisation of the activities.

5. Being able to predict change propagation more than one step ahead is beneficial but yields diminishing returns. This implies that, under the assumptions of the simulation, change prediction techniques involving substantial knowledge of particular cases does not significantly assist in change scheduling – although it may be beneficial for other reasons, for instance in deciding how to implement the change to avoid propagation. Such effects are not studied by our simulation.

6.2 Implications

The analysis of Section 6.1 highlights the potential impact of change prediction research to enhance future design environments. In particular, it highlights the importance of case-specific change prediction and, notably, the potential benefits to be gained beyond the statistical change prediction techniques reviewed in Section 2.2. The findings thus suggest that further research in change prediction could focus on the use of specific project data to support change prediction, and consider how change predictions could be best presented to assist project managers in scheduling change.
Considering these issues in the context of future design environments, information such as change histories and meta-data regarding the maturity and provenance of design information will be more readily available in the systems currently being developed across the aerospace industry. We argue that this maturity meta-data could be used to automatically populate detailed models that could support more targeted, case-specific change prediction. This is our main area for further work leading on from the research presented here.

7. Discussion and future work

7.1 Improving the model

The simulation approach used in this paper to explore the impact of change prediction capability on process efficiency has not been validated, yet has reasonable face validity. The change model of the helicopter is representative of many complex aerospace products, and was created and validated through substantial industrial involvement. The simulation algorithm is also based on an academically-accepted and empirically-supported model of how change propagates through a design (i.e., CPM), and does not require any input data beyond that provided by this model.
Nevertheless the simulation described in this paper is only a starting point, and there are thus many opportunities to improve and validate the work presented here. These include:

1. **Incorporating change packaging.** It seems likely that engineers would be able to foresee some major cycles of change and package them into a single set of tasks to avoid unnecessarily revisiting nodes; perhaps only larger cycles or those crossing significant organisational boundaries would remain undetected.
2. **Modelling the real cost of rework.** The model would more accurately reflect the change implementation process if the real cost of change implementation were modelled rather than simply the number of steps. This could be related to the time required to complete rework on a component according to the type and magnitude of the change.
3. **Modelling multiple initiating components.** Changes do not occur in isolation but in practice are affected by other changes to the same components occurring concurrently, and by the ongoing design process. Giffin et al. (2008) discuss how the largest change networks they identified arose from merging of the effects of change propagating from multiple initiating components. Multiple concurrent changes, and the availability of multiple personnel to execute these changes, are a significant complicating factor of change processes in practice and thus should be incorporated in the model to better reflect reality.
4. **Validation.** To validate the simulation, the statistical properties of the change trees generated could be compared to those of real change databases obtained from companies, such as the Giffin dataset. Such databases capture the history of how a change propagates.

7.2 Applying the model

The experiments performed so far consider only a few aspects of change scheduling. However, with further development the simulation approach presented in this paper could provide a testbed to explore many aspects of change processes and their coordination. For instance, coordination might also consider the impact or duration of the tasks to be performed – perhaps change affects certain elements more profoundly than others and reworking them has a greater impact (in cost, time, resources, etc.) than on others. Redoing a specific activity may be discouraged because it involves interaction with third-party organizations (who might charge setup or other fees). A related question and a simulation limitation is that, in reality, designers do not only choose which components to rework, but may decide how to rework a component in such a way as to prevent or minimise the potential for propagation to other components. Applying and extending the model to explore such issues is another opportunity for further work.

Future working environments which cross organisational and workflow boundaries will increase still further the complexities of change propagation and scheduling, as multiple pending tasks which are not immediately or obviously related may compete for attention. Conversely, such environments may also help scheduling, since capabilities such as automatic model creation and the retention of workflow and data changes should make many workflows more predictable. Understanding how to account for these complexities in managing change processes will be another of our areas for research in future.

8. Conclusion

Future design environments in aerospace and elsewhere will necessitate improved management of the propagation and impacts of changes. To ascertain whether change prediction can assist in making better work prioritisation decisions, this paper has developed a new simulation approach and applied it to a model of a complex aerospace product, which was elicited from industry. We used an accepted technique to generate potential change propagation trees and applied Monte Carlo methods to generate a sample space within which multiple scheduling policies could be evaluated and compared. The experiments revealed that poor coordination of change activity can result in significant process inefficiencies, that the potential for inefficiency increases for larger change networks, and that a modest ability to accurately predict change propagation *in the specific case at hand* could have a dramatic effect in reducing unnecessary rework. Unexpectedly, the experiments also suggested that the capability of predicting multiple steps of change propagation would provide only minimal additional improvement.

References

Clarkson, P.J., Simons, C.S., Eckert, C.M., "Predicting change propagation in complex design", Transactions - American Society of Mechanical Engineers Journal of Mechanical Design, Vol.126, No.5, 2004, pp 788-797.

Eckert, C.M., Clarkson, P.J., Zanker, W., "Change and customisation in complex engineering domains", Research in Engineering Design, Vol. 15, No.1, 2004, pp 1-21.

Ford, D. N. and Sterman, J. D., "Dynamic Modeling of Product Development Processes", System Dynamics Review, Vol. 14, 1998, pp. 31-38.

Fricke, E., Gebhard, B., Negele, H., Igenbergs, E., "Coping with Changes: Causes, Findings, and Strategies", Systems Engineering, Vol.3, No.4, 2000, pp 169-179.

Giffin, M., Weck, O., Bounova, G., Keller, R., Eckert, C., Clarkson, P.J., "Change Propagation Analysis in Complex Technical Systems", Journal of Mechanical Design, Vol.131, No.8, 2009.

O'Brien, C., Smith, S.J.E, "Design maturity assessment for concurrent engineering co-ordination", International Journal of Production Economics, Vol. 41, No.1-3, 1995, pp 311-320.

Rouibah, K., Caskey, K.R., "Change management in concurrent engineering from a parameter perspective", Computers in Industry, Vol. 50, No.1, 2003, pp 15-34.

Rowell, W.F., Duffy, A.H.B., Boyle, I.M., Masson, N., "The nature of engineering change in a complex product development cycle", 7th Annual Conference on Systems Engineering Research 2009 (CSER 2009), R.S. Kalawsky (Ed.), available at http://cser.lboro.ac.uk/abstracts/S04-25.html.

Rutka, A., Guenov, M.D., Lemmens, Y., Schmidt-Schäffer, T., Coleman, P., Rivière, A., "Methods for Engineering Change Propagation Analysis", Proceedings of the 25th International Congress of the Aeronautical Sciences,Hamburg, 2006, available from http://hdl.handle.net/1826/2621
Zhao, Z-Y., Lv, Qian-Lei, You, W-Y., "Applying Dependency Structure Matrix and Monte Carlo simulation to predict change in construction project", Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, IEEE, 2008, Vol.2, pp 670-675.

Dr. David C. Wynn
Senior Research Associate
Engineering Design Centre, University of Cambridge. UK.
Telephone: +44.1223.748565
Email: dcw24@cam.ac.uk