

Change Propagation Analysis in Complex Technical Systems

Monica Giffin

Olivier de Weck¹
e-mail: deweck@mit.edu

Gergana Bounova

Engineering Systems Division,
Massachusetts Institute of Technology,
Cambridge, MA 02139

Rene Keller

Claudia Eckert

P. John Clarkson

Engineering Design Centre,
University of Cambridge,
UK

Understanding how and why changes propagate during engineering design is critical because most products and systems emerge from predecessors and not through clean sheet design. This paper examines a large data set from industry including 41,500 change requests that were generated during the design of a complex sensor system spanning a period of 8 years. In particular, the networks of connected parent, child, and sibling changes are resolved over time and mapped to 46 subsystem areas of the sensor system. These change networks are then decomposed into one-, two-, and three-node motifs as the fundamental building blocks of change activity. A statistical analysis suggests that only about half (48.2%) of all proposed changes were actually implemented and that some motifs occur much more frequently than others. Furthermore, a set of indices is developed to help classify areas of the system as acceptors or reflectors of change and a normalized change propagation index shows the relative strength of each area on the absorber-multiplier spectrum between -1 and $+1$. Multipliers are good candidates for more focused change management. Another interesting finding is the quantitative confirmation of the “ripple” change pattern previously proposed. Unlike the earlier prediction, however, it was found that the peak of cyclical change activity occurred late in the program driven by rework discovered during systems integration and functional testing. [DOI: 10.1115/1.3149847]

1 Introduction

Over the past decade both industrial and academic interests in change propagation have risen, yielding some small-scale in-depth studies as well as a variety of tools aimed at aiding analysis and prediction of change propagation. Understanding how and why changes propagate during the engineering design process is critical because most products and systems emerge from predecessors and not through clean sheet design. Even a de novo (clean sheet) design is often subject to changes later in its development process. Additionally, product development schedules, costs, and quality are driven to a large extent by change and rework activities [1]. This paper develops a new network-based analysis technique and applies a number of existing change propagation analysis methods to a large data set from industry.

Following a discussion of prior literature on change propagation and change management we describe the system that was investigated in this research. Next, we explain the data mining procedure we followed and give an overview of the data set containing over 41,500 change requests. To our knowledge this data set is the largest that has been investigated and published in the design literature. To extract insights from the data we develop change motifs as well as a number of indices to measure the propensity of a subsystem area to accept, repel, or propagate changes. A change *motif* is a simple pattern of connected change requests from which more complex change networks emerge.

A number of previously developed change propagation analysis methods such as change design structure matrices (Delta-DSMs) and the change propagation method (CPM) were also applied to the data set. Our focus is on a posteriori analysis of change activity on a large real-world design project that recently concluded after an 8 year development period. The purpose of the research

was to gain insights into the design process of a new version of a previously successful system. In doing so we ask the following questions.

- (1) What are some significant statistical observations of the distribution of change requests in terms of their likelihood of implementation and magnitude of change?
- (2) Are most change requests isolates or do they occur in larger sets of connected changes?
- (3) If larger networks of changes occur, do they originate from a single instigating parent change or are the actual patterns of change propagation more complex?
- (4) Are changes distributed evenly or unevenly throughout the system and if so, do some areas act as multipliers or absorbers of changes?
- (5) Is the amount of activity, i.e., the rate at which new change requests are generated, constant over time or can we observe patterns over the course of the project?
- (6) What implications for the choice of product architecture and design change process management can be derived from the analysis, if any?

In order to answer these questions, we first provide a brief literature review in the area of change propagation and management from which flow the research objectives and approach that was followed in this paper. Next, we describe the system that was the subject of redesign, the data mining procedures and preliminary results of a statistical analysis. The core of this paper consists of the network analysis of change requests using a combination of graph theory and pattern (motif) analysis. We then develop a set of indices that allow quantification of change activity by subsystem area. In particular, the normalized change propagation index (CPI) shows the relative strength of each area on the absorber-multiplier spectrum between -1 and $+1$. Another interesting finding is the quantitative confirmation of the “ripple” change pattern previously proposed by Eckert et al. [2]. We conclude this paper with a summary of our findings and recommendations for future work.

¹Corresponding author.

Contributed by the Design Theory and Methodology Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received June 16, 2007; final manuscript received March 24, 2009; published online July 9, 2009. Review conducted by John K. Gershenson. Paper presented at the ASME 2007 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2007), Las Vegas, NV, Sept. 4–7, 2007.

2 Literature Review

Similar to others [2] we define change propagation as the process by which a “change to one part or element of an existing system configuration or design results in one or more additional changes to the system, when those changes would not have otherwise been required.” An understanding of change propagation in engineering systems is important in order to design, manufacture, and operate those systems on schedule and within budget. A recent study by the Aberdeen Group [3] showed that the majority of changes—although necessary for innovation—cause “scrap, wasted inventory, and disruption to supply and manufacturing.” The report also shows that most companies do not properly assess the consequences of changes. Only 11% of all companies were able to “provide a precise list of items affected by a change” in the development of a single product, while only 12% were able to assess the consequences of changes on the life cycle of the product. On the other hand, the majority (82%) of companies interviewed also stated that they are mostly concerned about increasing product revenue, which leads to innovation and in turn to the introduction of changes to their existing products.

Change propagation research and literature draw from change management, engineering design, product development, complexity, and graph theory and design for flexibility. The interdisciplinary nature of the work leads to consideration of industrial contexts whenever and wherever possible. Impacts of change propagation are documented by the pervasive configuration management systems of industry, see Refs. [4,5]. Within the literature directly concerning change propagation, there are several main engineering themes:

1. Descriptions of the nature of change propagation, which state the reasons for interest in the field,
2. results of studies, mainly in small sets of data (usually fewer than 100 changes),
3. development of tools for predicting change and visualizing networks of changes, and
4. methods for controlling change propagation through better design decisions. This includes work in “design for changeability” [6], which emphasizes designing products and systems in the first place such that future changes are either not needed or are limited in terms of their extent.

The majority of the work to date has been to define and characterize engineering change propagation as in Ref. [2]. They identified engineering changes as resulting from two main categories: *initiated changes* and *emergent changes*. Initiated changes are those intended by a stakeholder, while emergent change is unintended and occurs when some aspect of the system design requires changing because of errors or undesirable emerging system properties, often due to an earlier initiated change. Change and configuration management is a critical prerequisite for investigations into change propagation. However, existing standards of configuration management (including DOD 5015.2) [7] are geared toward minimal record keeping, whereas investigating change propagation requires a more comprehensive recording of change requests and their disposition over time [8]. Wright’s [4] survey defined an engineering change as the modification of a component or product, which has already entered production. Wright [4] recognized that in that context engineering changes are seen as “evil, foisted on the manufacturing function by design engineers who probably made a mistake in the first place.” In contrast, engineering changes may in reality be the vehicle by which firms maintain or grow market share—it is rare indeed to find a firm that does not use incremental innovation. The largest deficiency noted in the engineering change management literature as of Wright’s publication was the lack of research addressing the effects of engineering change in the incremental product development process.

Additional studies of engineering change management in situ have been conducted in three Swedish engineering companies by

Pikosz and Malmqvist [5], leading them to suggest strategies for improving change management practices. Huang and Mak [9] surveyed a significant cross section of UK manufacturing companies regarding their engineering change management practices. Following a case study at Westland Helicopters Ltd., Eckert et al. [2] defined different components with regard to change propagation as falling into several categories, paraphrased here:

- a. *constants*: components unaffected by change
- b. *absorbers*: absorb more changes than they cause
- c. *carriers*: absorb and cause a similar number of changes
- d. *multipliers*: generate more changes than they absorb

As follow-on, Clarkson and Eckert together with Simons [10] described a method for predicting change propagation paths using DSMs. When coupled with measures of likelihood that changes will, in fact, propagate, a potential outcome is a scaled rating of the likely impact of a given change. Keller et al. [11] also begin to address visualization using their CPM tool. This CPM tool was tested in this paper on the large data set described below. Change propagation can be analyzed for individual products as well as potentially for product platforms [6,12].

Building on previous work [13], addresses the area of system design for flexibility with the specific goal of minimizing change propagation through preventative measures (e.g., splitting multipliers into subcomponents). By modeling the propagation of changes in functional requirements of automotive platforms to physical components, those parts of the system were identified that could benefit from flexibility. They introduce a change propagation index (CPI) as follows:

$$CPI_i = \sum_{j=1}^N \Delta E_{j,i} - \sum_{k=1}^N \Delta E_{i,k} = \Delta E_{out,i} - \Delta E_{in,i} \quad (1)$$

where N is the number of elements or areas in the system and $\Delta E_{i,j}$ is a binary matrix (0,1) indicating whether the i th element is changed because of a change in element j . CPI helps classify elements as multipliers ($CPI > 0$), carriers ($CPI = 0$), and absorbers ($CPI < 0$). Moreover, they emphasize that not only the number of changes are important but also their magnitude in terms of *switching costs*, i.e., the aggregate cost of switching the configuration of a system from one state to another. This original definition of CPI is modified and refined in this paper.

Previous work on change data sets has included a study of approximately 100 changes during a 4 month study by Terwiesch and Loch [14], who were involved in the change process for an automobile climate control system. There are a number of reasons why research on more complete data mining for large scale engineering changes has been lacking to date.

- (a) Firms typically focus on tracking changes in ongoing projects, relegating the information at completion to the historical record rather than analysis.
- (b) Changes are often due to oversights and errors and there may be no incentive in exposing flawed or less than ideal processes to a wider audience.
- (c) The data sets may be very large, saved in heterogeneous formats, and generally difficult to access, mine, analyze, and visualize.
- (d) The ability to manage change effectively is competition sensitive.

This paper provides a unique opportunity for examining the approximately 41,500 change records of a large project, where changes were recorded consistently over a period of 8 years, with the first author [8] having intimate knowledge of the system development and change management over more than half of that time period.

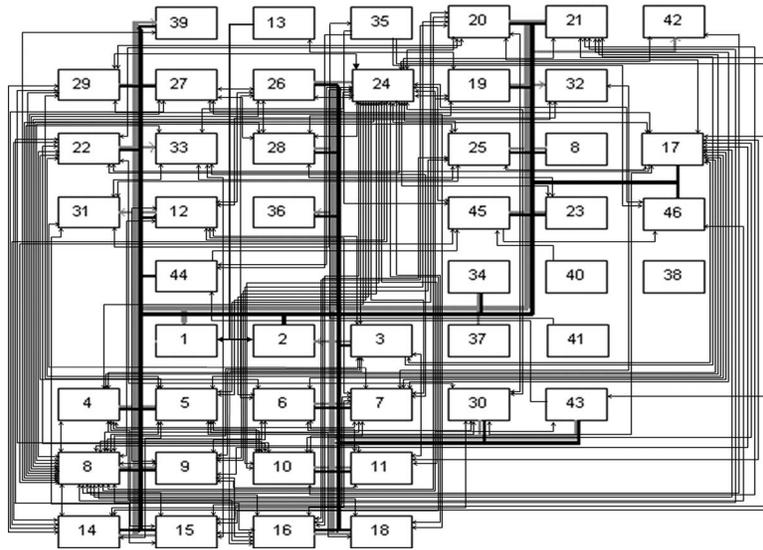


Fig. 1 System block diagram (46 areas)

3 System Description

The data for this research are the result of a large technical program performed on a United States government contract, with multiple stakeholders, and involving globally distributed complex hardware and software subsystems and interactions, as well as distributed users and operators. The system can be broadly described as a large scale sensor system that was developed as part of a billion-dollar class project. The system was derived from an earlier generation. Figure 1 shows a diagram representing the connections between major subsystems (“areas”).

The system could be decomposed into distinct areas consisting of *software*, *hardware*, and *documentation*. Software accounted for the vast majority of the undertaking; therefore, we would expect changes to the software to be the most prominent set of records in the data set, followed by changes to software-related documentation (including requirements, design, training, and operating documentation). The hardware segment of the program required very limited development of new components. The vast majority was to be reused from a pre-existing system, thanks to the careful introduction of a buffer component between the reused hardware design and the new software.

The system map (Fig. 1) was derived from the detailed design phase artifacts describing the various subsystems and designed interactions and is representative of a common format for those artifacts. Software interactions mainly consisted of information and data transfer. An initial evaluation identified 46 areas with the potential to be affected by proposed changes. These include software components, different levels of documentation, and hardware. The term area will be used throughout this paper, and may be thought of as a coherent segment, perhaps analogous to a subsystem.

4 Data Set

The data set evaluated in this paper consists of more than 41,500 proposed changes—technical, managerial, and procedural—over the course of eight calendar years. In contrast to some previous work, the definition of change is used to refer to any changes made after initial design (this is more in keeping with Cohen and Fulton’s [15] view of engineering changes, and expanded to include the nontechnical changes which often cause or accompany the traditional “engineering changes”). In later portions of the data, the character becomes more similar to the definition used by Wright [4] “... a version of the product in use by customers is changed and the new version is released back to the

customers.” Thus the data set spans the design phase, integration phase, and initial operations. The data were saved in a company-developed configuration management system. A noteworthy aspect of this data set, in addition to its size, is the method of change tracking, which was used. Contrary to common practice in the industry, all changes resulting from a single problem were not tracked under the same identifier. Instead, *unique identifiers were used for changes on a per-area basis*.

In order to obtain a manageable data set from the original change requests archived in the change management database, several processes were used. First, a subset of each change request (CR) was written to a text file. Items such as specific build identifiers for software changes, and location codes were omitted here. Next, the text file thus generated was parsed and written into a MySQL database using a PERL script, see Ref. [8] (Appendix A) for specifics. Each record present in the text file was preserved across this transition. With all of the data in a manageable format (the text file would have been upwards of 120,000 pages if merged into a single Microsoft Word document) the tasks of sorting, categorizing, and filtering the data became possible.

Next, the names of 496 individuals found to be mentioned in the entries were anonymized as a numbered engineer or administrator identifier. Following the considerable effort of extracting and filtering the data, a simplified change record format was written out via another PERL script. Each data entry preserves the original change request identifier, and is printed in the same format, as shown in Table 1.

Table 1 Typical CR record

ID number	12345
Date created	06-MAR-Y6
Date last updated	10-JAN-Y6
Area affected	19
Change magnitude	3
Parent ID	8648
Children ID(s)	15678, 16789
Sibling ID(s)	9728
Submitter	eng231
Assignees	eng008, eng231, eng018
Associated individuals	admin001, eng271
Stage originated, defect reason	[blank], [blank]
Severity	[blank]
Completed?	1

Table 2 Change magnitude classification by software lines of code (SLOC) and work hours

5	Total SLOC ≥ 1000	≥ 200 total hours
4	$200 \leq \text{total SLOC} < 1000$	$80 \leq \text{total hours} < 200$
3	$50 \leq \text{total SLOC} < 200$	$40 \leq \text{total hours} < 80$
2	$10 \leq \text{total SLOC} < 50$	$8 \leq \text{total hours} < 40$
1	$1 \leq \text{total SLOC} < 10$	$1 \leq \text{total hours} < 8$
0	Total SLOC < 1	Total hours < 1

- (i) *ID numbers* were assigned based on when the change request was first submitted to the electronic database (*Date created*). A total of 41,551 entries were assigned IDs. *Date last updated* notes the last time anything was altered for that particular change request.
- (ii) *Area affected* describes the segment of the system affected. As described above, 46 areas were identified (see Fig. 1).
- (iii) *Change magnitude* is a categorization of the anticipated impact of the request. Magnitudes were recorded as 0 to 5 (or -1 if no information was present), as shown by Table 2:
- (iv) The *parent*, *children*, and *sibling* fields identify other associated change request records by ID number. A child is a direct result of a parent (typically the parent is the original problem description, while the child is one of a set of changes required to correct the problem stated in the parent). Siblings may either be children of the same parent, or otherwise related. Siblings may often supersede one another or may be complementary changes in different areas required to realize a common goal. Figure 2 shows the sample patterns of change requests. Parent-child (pc) relationships are shown as solid, unidirectional arrows, while sibling relationships are shown as dashed, bidirectional arrows. Gray circles show change requests that have been proposed, but are still unresolved, black circles are completed (implemented) changes, while circles with an "X" show disapproved (rejected) changes.
- (v) *Individuals* who were instrumental in processing of a change request are indicated by the submitter, assignees, and associated individuals fields.
- (vi) *Stage originated*, *defect reason*, and *severity* are all fields from the original change request portraying whether a change request was made as a direct result of a documented customer request. Unfortunately information in these fields was not always entered consistently.
- (vii) The last field, *Completed?* indicates the final status of the change request at the time we extracted the database. A value of 1 indicates that the original change request was

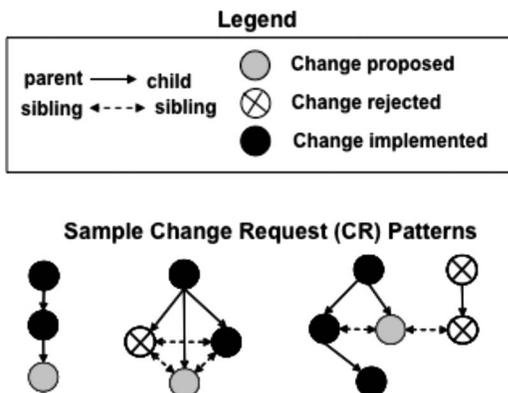


Fig. 2 Parent-child and sibling change relationships

Table 3 Change request status by magnitude

	Total No.	Completed	Unresolved/in process	Withdrawn/superseded/disapproved
5	124	120 (96.7%)	0	4 (3.3%)
4	751	724 (96.4%)	3 (0.4%)	24 (3.2%)
3	2,049	1942 (94.8%)	11 (0.5%)	96 (4.7%)
2	5,295	4918 (92.8%)	14 (0.3%)	363 (6.9%)
1	7,430	6937 (93.3%)	58 (0.8%)	435 (5.9%)
0	10,437	5323 (51%)	255 (2.4%)	4859 (46.6%)

completed. A value of 0 indicates that it was still in process as of data set capture. A value of -1 indicates that the change request was withdrawn, disapproved or superseded by another change request.

5 Initial Analysis

Of the 41,551 change request records in the data set, change magnitudes could not be determined for 15,465 of them. The 26,086, which could be evaluated from the provided data, were distributed, as shown in Table 3 and Fig. 3.

Figure 3 shows the distribution of change magnitude in terms of the total number and status of change requests. Two key observations can be made.

1. There is an inverse relationship between expected change magnitude and its frequency of occurrence. Very large changes are rather infrequent while small changes, on the other hand, are commonplace.
2. Nearly half the smallest (magnitude 0) changes were either withdrawn, superseded, or disapproved (46.6%), whereas the majority of the larger changes (magnitudes 1-5) were approved and carried out to completion (between 92.8% and 96.7%).

6 Change Propagation Analysis

The initial analysis shows that change was, in fact, happening and tracked at an aggregate level. However, it is necessary to explore how those changes associated with the system affected different areas. Did they occur in expected areas? Were they uniformly distributed throughout the system?

6.1 Change Design Structure Matrix. In order to better understand to what extent changes propagating through the system followed expected connections, we first generated a structural design structure matrix (DSM) [16] in Fig. 4. The DSM documents direct interconnections between areas including physical connections, as well as information and energy flows. This format cap-

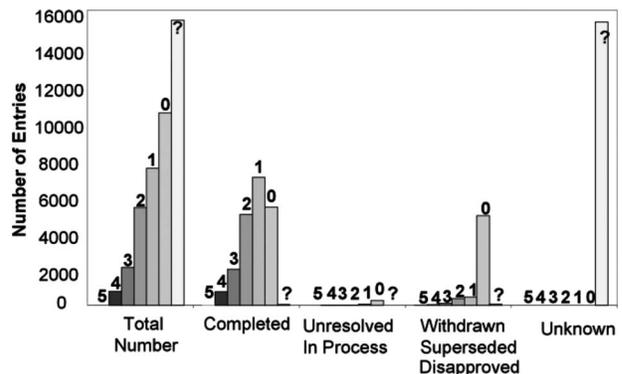


Fig. 3 Change magnitude distribution

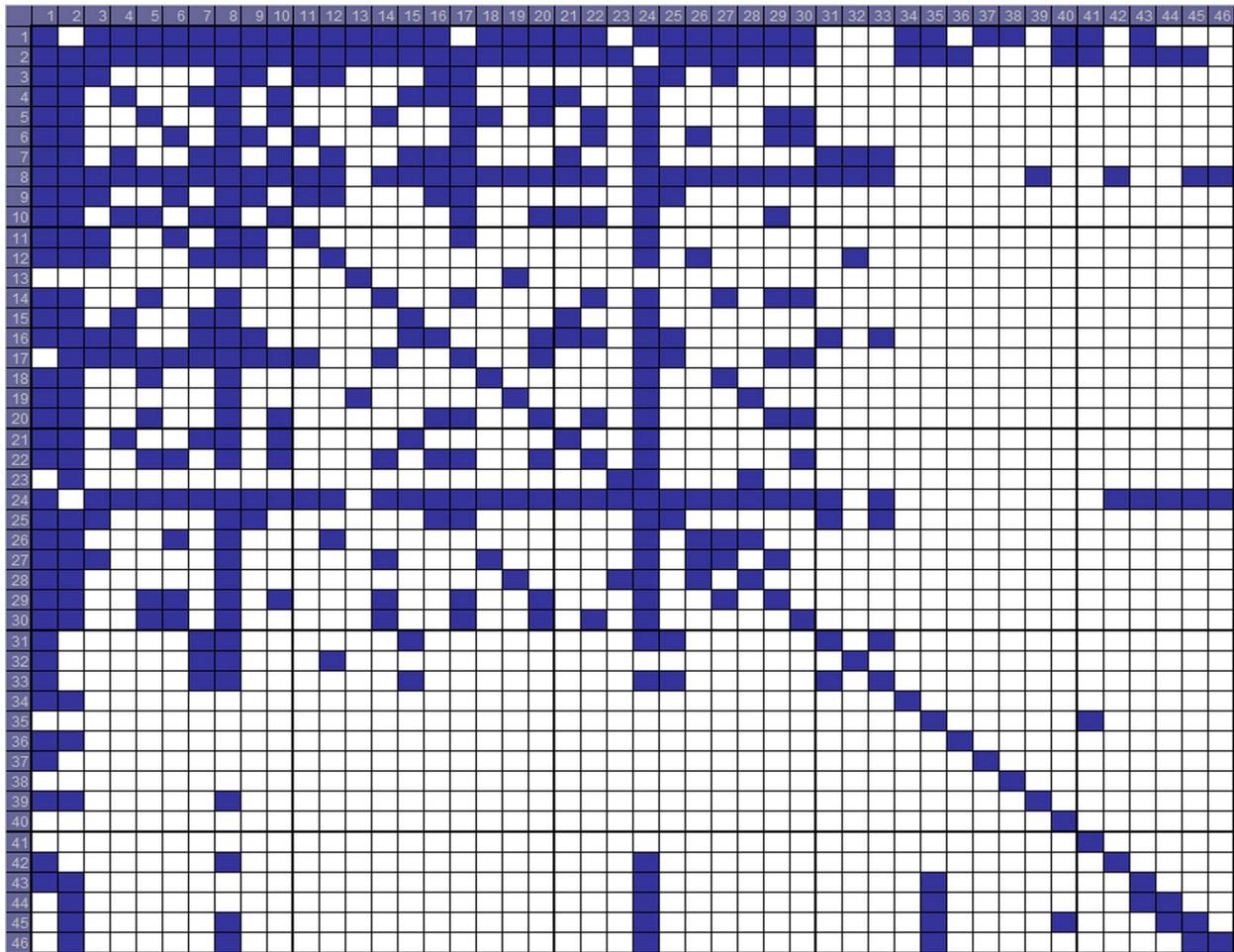


Fig. 4 System structural DSM (ordered 1–46)

tures the expected flow of changes between “adjacent” areas as they were defined in the detailed design information (see system map Fig. 1).

The DSM is nearly (but not perfectly) symmetric and shows the direct interconnections between adjacent areas in the system map. In order to create a map of how changes actually affected the system, we needed to derive what we will call a change DSM or Δ DSM (see Ref. [17]). In this version of Δ DSM column headings show what Clarkson et al. [10] call the “instigating” area, and rows show the “affected” area in line with DSM convention of mapping out effects. As an example we may have a change request for Area 5 originating from a change request in Area 1 as its parent and a change request for Area 8 as its sibling. One interpretation is that the Area 1 change propagated to Area 5 and most likely to Area 8 as well. However, it is possible that when we look at the dates and completion information for the change request for Area 8 (see format in Table 1) we will find that it had a comple-

tion of -1 , and was closed before the change to Area 5 was opened, in which case the better conclusion is that a change was made to Area 5 instead of Area 8 to finish resolving the problem, which caused the original change request in Area 1. These relationships are captured in the Δ DSM. The subtlety between an actual propagated change and a substituted change is illustrated in Fig. 5.

A listing of individual edges was then created, along with an area by area summary. Each edge indicates either a parent-child or sibling relationship that was recorded between two change requests (see Fig. 2). These data allowed aggregate frequency and impact values to be calculated for changes propagating within and between areas of the system. The frequencies of propagation were computed for the edges yielding a matrix from which an excerpt (for the first 25,000 entries) is shown in Table 4.

Note that depending on the nature of the area, there might be no *internal* propagation (emergent changes) or they might be very

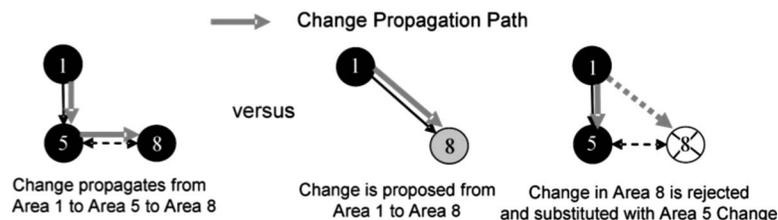


Fig. 5 Propagated change versus substituted change

Table 4 Change propagation frequency matrix (6×6 excerpt)

	Area					
	1	2	3	4	5	6
1	0.4843	0.0011	0.0136	0.0057	0.0125	0.0023
2	0.0061	0.0000	0.0000	0.0030	0.0000	0.0000
3	0.0173	0.0000	0.1053	0.0050	0.0012	0.0000
4	0.0224	0.0000	0.0112	0.0449	0.0000	0.0000
5	0.0137	0.0000	0.0000	0.0000	0.1262	0.0000
6	0.0417	0.0000	0.0000	0.0000	0.0000	0.0833

common. Area 1, for example, contains the requirement documentation, both at the program level and at the software and hardware item levels. Nearly half of the time (0.4843) that requirements changes were made within this area, they were associated with another requirements change (also within Area 1). A change in Area 1 caused changes in Area 6 with a frequency of 4.17%.

This allowed the construction of a change DSM from all 41,551 change records, see Fig. 6. The DSM view in Fig. 6 is the result of overlaying the structural DSM in Fig. 4 with the propagation frequency (Table 4) for all change requests.

The dark matrix cells labeled “P” are those where a structural connection was known from the DSM shown in Fig. 4 (thus change propagation was predicted) and change propagation actually occurred. The lighter cells labeled “S” are those areas where change propagation (to the fourth decimal place) did not occur,

yet a structural connection was present. The areas labeled “C” highlight those places where a direct structural connection was not noted, but change propagation did occur. This was an unexpected finding since change propagation was previously assumed to follow only along structural links. In the full 41,551 data set, the frequency of change only exceeded 10% propagation in just over $2\frac{1}{2}\%$ of the potential cases. This is particularly interesting in relationship to previous work focused on mechanical systems, where reported change propagation frequencies were generally larger. In this software-dominated system, change propagation was less clear cut. Our hypothesis is that the architecture of this system was carefully crafted to be modular from the start, thus leading to less frequent interarea change propagation compared with a purely mechanical system or to nonmodular software. The other key takeaway message is that changes can indeed propagate between areas that seem to not be directly connected based on the system structural DSM alone (see Figs. 1 and 4).

6.2 Change Networks. Graph theory was also used to analyze the data set and patterns emerged from the data as analysis proceeded. Initially, a search was performed for the largest change component (set of mutually linked changes through a combination of parent-child and sibling changes). A search revealed a largest component of 2566 linked changes (see Fig. 7), while the majority of the components found were comprised of fewer than 10 changes. Table 5 shows the size of the five largest networks of connected changes in the project. Given the size of the largest network, it was realized that the goal of analyzing several of the

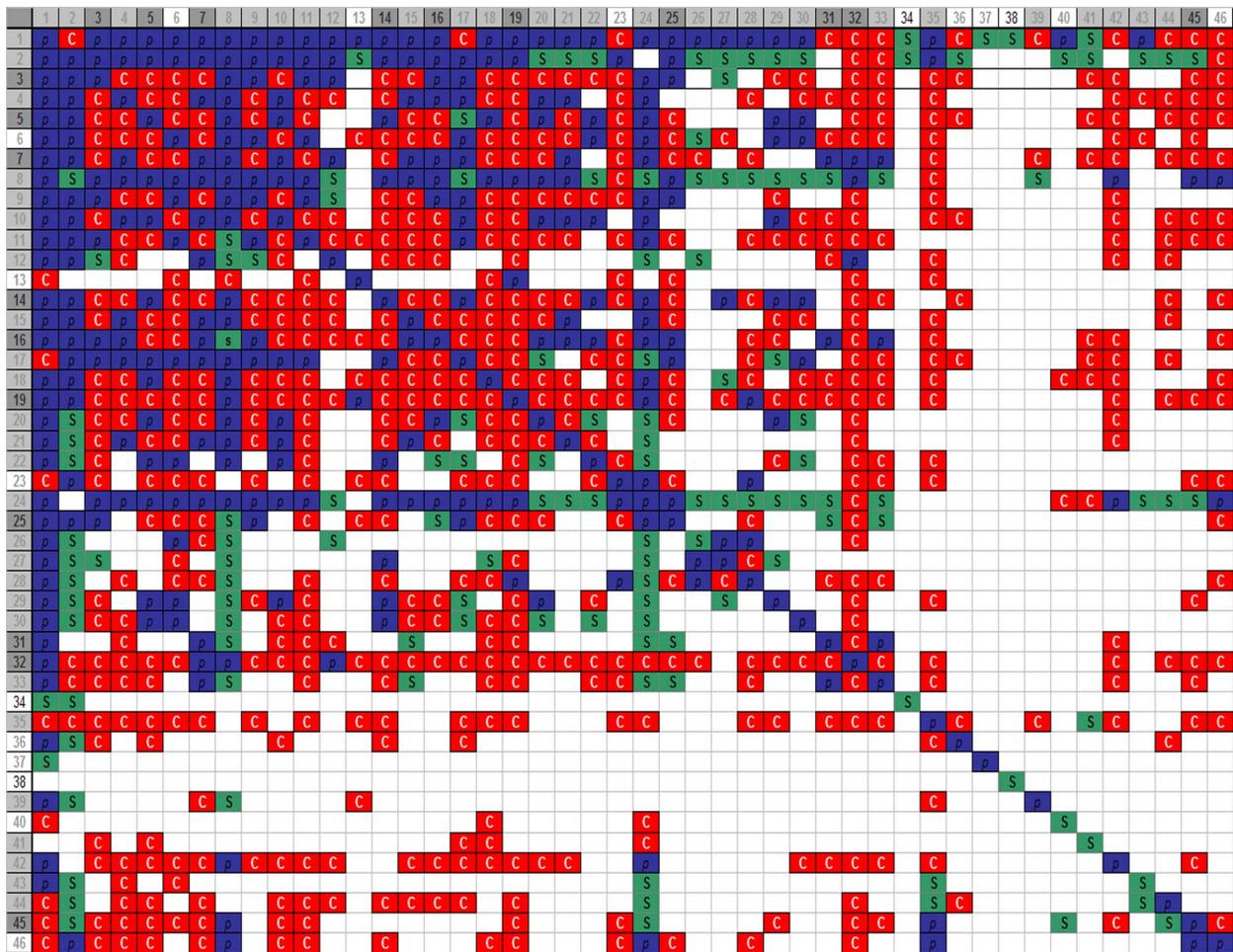


Fig. 6 Overlay of baseline DSM (Fig. 4) with change propagation frequency matrix of actually implemented changes (Table 4)



Fig. 7 Largest change network: 2566 related changes

largest network components would be beneficial, but that one might gain more insight by initially analyzing smaller instances of change networks.

One question we did seek to answer was whether or not the large change networks did originate from a *single parent change request* and then grew through multistep change propagation from this initial nucleus. What we found was that this was *not* the case. The large change networks tended to coalesce based on nucleation in different parts of the system, see Fig. 8, which shows the growth of the fourth largest change network with 87 related change requests (87CR) at four different snapshots in time between years 5 and 8.

One possible mechanism for this phenomenon is that a set of problems during system design was initially diagnosed as a set of unrelated issues in different areas. With further maturity of the design, these issues were then recognized as being related to each other and a “superchange” was created to fix all problems at once, or alternatively, a common root cause was identified, giving rise to a larger connected network of changes. Through the change network analysis we found that change propagation in large technical systems is actually much more complex than we thought initially. One promising approach to this is to develop a set of basic change propagation patterns (“motifs”) and to analyze their frequency of occurrence in the overall data set as well as in the individual change networks (see Table 5). This is the subject of Sec. 7.

7 Change Patterns (Motifs)

Motifs are the basic building blocks of larger change networks. Starting with a relatively low-order representative 11-change network we first define 1-, 2- and a subset of 3-change motifs. These motifs represent the clearly defined building blocks, which allow analysis of the more complex change networks such as the 87-CR component in Fig. 8 and even the 2566-CR network in Fig. 7. We begin by considering the network of 11 related changes shown in Fig. 9.

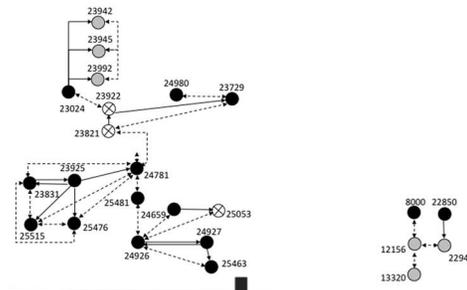
Table 5 Largest change networks

Rank	Connected changes
1	2566
2	445
3	170
4	87
5	64

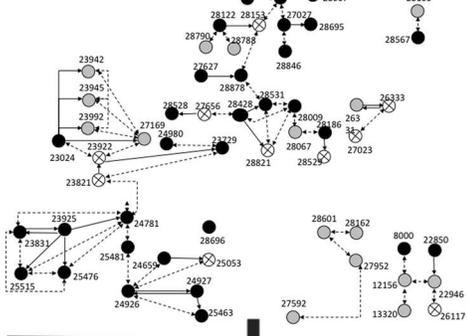
1: Year 5 new CRs
ID 1-22000



2: Year 6 new CRs
ID 22000-26000



3: Year 7 new CRs
ID 26000-29000



4: Year 8 updated
final status of CRs

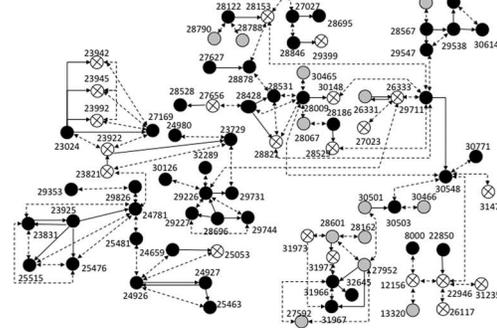


Fig. 8 Time-lapse analysis (in four increments from top left to lower right) for the fourth largest change network with 87 change requests

Each node in Fig. 9 represents a change request identified by a unique identification number (e.g., 32822), which corresponds to the sequential chronology when the change request was initially generated. At the end of the program, each change request was in one of the three states: proposed (meaning it was still unresolved), rejected, or implemented. Based on the data saved for each change

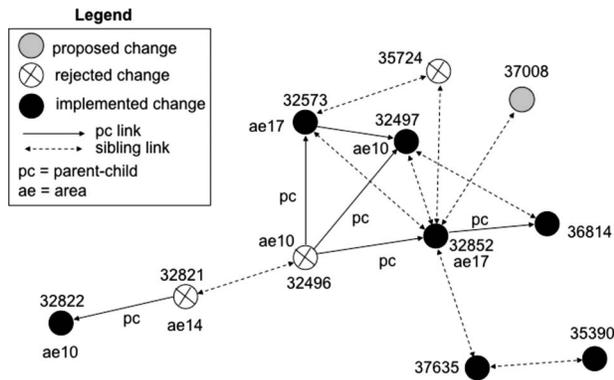


Fig. 9 Change network with 11 connected change requests (11-CR) from the larger data set. The initial change request is 32,496 (rejected) in the middle of the graph.

request (see format in Table 1), we can identify which other changes are either a parent, child, or sibling to the particular change request in question. Consider, for example, change request 32496 in Fig. 9. It is a sibling to 32821, but a parent to 32573, 32497, and 32852, respectively. Parent-child (Pc) links are shown as solid directed arrows, while sibling relationships are shown as bidirectional dashed arrows. For those change requests involved in parent-child relationships, we also identify their associated area, e.g., change requests 32822, 32496, and 32497 all relate to Area 10 of the system (see Fig. 9).

7.1 1-Motifs. The simplest consideration is that of 1-motifs, i.e., looking at one change request at a time. Figure 10 shows the possible patterns. A change always starts as a proposed change (in step i), and either remains open or is ultimately implemented or rejected at some later step j. We use C00, C11, and C10 to designate these three motifs.

An analysis of the frequency of occurrence of these patterns in the 11-CR network of Fig. 9 is shown in Table 6. We see that 63.6% of all changes were implemented and nearly one in four changes (27.3%) was rejected. We will examine the same statistics for the entire data set below.

7.2 2-Motifs. The 1-motifs do not take into account how change requests relate to each other. To examine this we define 2-motifs, which come in two families, depending on whether parent-child (Fig. 11) or sibling changes (Fig. 12) are involved.

In the parent-child 2-motif family we have nine unique patterns, which are designated as *PXY*. The *P* identifies the pattern as a parent-child pattern, the number *X* expresses whether the parent

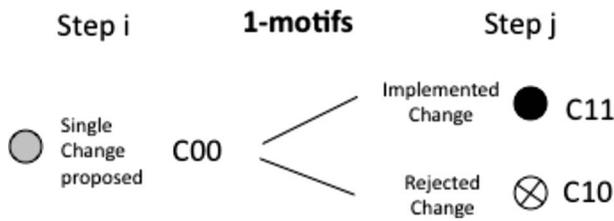


Fig. 10 1-motif change patterns

Table 6 1-motif analysis for 11-CR change request network

1-motif analysis	Proposed C00	Implemented C11	Rejected C10	Total
Comp 11	1 (9%)	7 (63.6%)	3 (27.3%)	11 (100%)

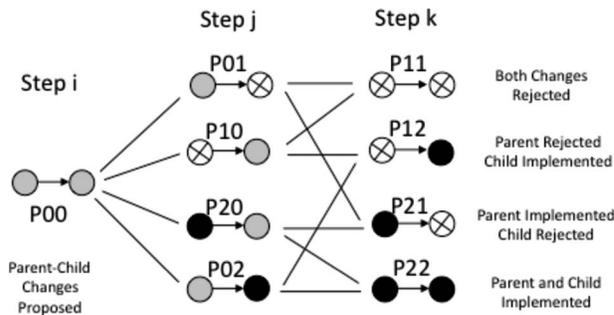


Fig. 11 Parent-child 2-motif family

change is proposed (0), rejected (1), or implemented (2), while the *Y* shows the status of the child changes as either proposed (0), rejected (1), or implemented (2).

The 2-motif sibling family, identified by *SXY*, only contains six unique patterns and is somewhat simpler than the parent-child 2-motifs due to the symmetry of the sibling relationship. By definition sibling changes are related, but one did not spawn the other. We can now search for the number of occurrences of each motif in a change network. Tables 7 and 8 show the results of the 2-motif occurrences in the simple 11-CR network of Fig. 9.

The results for the small 11-CR network suggest that some change motifs are much more frequent than others, but we have to inspect the entire data set to arrive at a definitive result.

7.3 3-Motifs. There are a number of possible 3-motifs, which represent “triangle” connections between change requests. The set of possible families is shown in Fig. 13. Note that a number of these families are deemed to be “illegal,” specifically those that

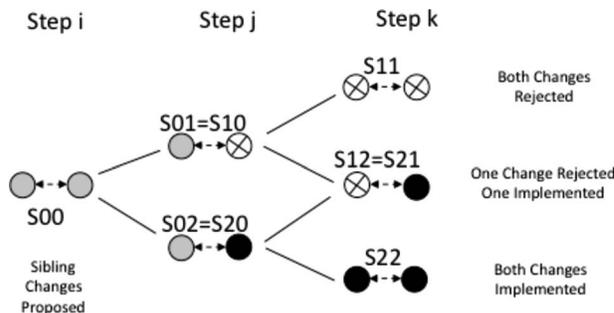


Fig. 12 Sibling 2-motif family

Table 7 2-motif statistics for parent-child changes in the 11-CR network of Fig. 9

	PC motifs									Total
	P00	P01	P10	P11	P02	P20	P21	P12	P22	
Comp 11	0	0	0	0	0	0	0	4 (80%)	1 (20%)	5 (100%)

Table 8 2-motif statistics for sibling changes in the 11-CR network of Fig. 9. (Sibling motifs are double counted due to their inherent symmetry. This can be double checked by inspection of Fig. 9.)

	Sibling motifs						Total
	S00	S01	S02	S11	S12	S22	
Comp 11	0	0	2 (10%)	2 (10%)	4 (20%)	12 (60%)	20 (100%)

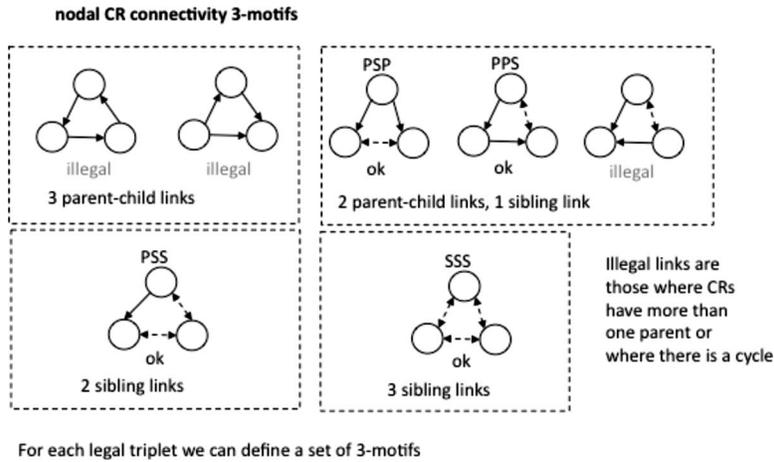


Fig. 13 Set of 3-motif change request families

form cycles and where one change request has more than one parent. Change propagation analysis should focus on the legal families, particularly the PSP, PPS, PSS, and SSS families.

Each of the legal families can be expanded, as shown for the PSP (parent-sibling-parent) family in Fig. 14. We can see that the combinatorial complexity of possible patterns is significantly larger than in the case of the 2-motifs.

The 3-motifs have potentially the most explanatory power, but are also the most involved to analyze for large networks. For example, the PSP222 motif can potentially be explained as follows: a parent change is proposed and in order to fully achieve the goal of the redesign two child changes are also required. The two child changes are related to each other as siblings and eventually all three changes are implemented. An analysis of the 11-CR net-

work in Fig. 9 in terms of 3-motifs reveals that a number of such patterns are present such as the PSP122 pattern involving change requests 32496, 32497, and 32852.

7.4 Change Acceptance and Reflection. An interesting question in the context of change analysis is whether some areas (subsystems and components) are generally accepting of change, or whether they tend to reflect change. To quantify this we define the following simple ratios, which we name *CAI* and *CRI*, respectively.

$$CAI_i = \frac{\text{total number of implemented changes in area } i}{\text{total number of changes originally proposed in area } i}$$

$$CRI_i = \frac{\text{total number of rejected changes in area } i}{\text{total number of changes originally proposed in area } i} \quad (2)$$

Note that each area of the system will potentially have a different value of *CAI* and *CRI* and that *CAI* and *CRI* may not necessarily sum to 1, since—as shown earlier—some change requests are still unresolved, even at the end of the project. A sample calculation for the 11-CR network in Fig. 9 yields the following results shown in Table 9.

This analysis allows a deeper understanding of which areas are more or less impacted by certain types of changes (the 11-CR network only impacts Areas 10 and 17 out of the 46 areas shown in Fig. 1) and which areas act as acceptors or reflectors of changes. All that is needed for this analysis is the 1-motifs (shown in Fig. 10). Section 8 shows the results of the analysis for the entire data set.

8 Change Impact Analysis for Entire Data Set

The 1-motif analysis results for the entire data set are shown in Table 10. We see that overall about half (48.2%) of all originally proposed changes were implemented, that 14.3% were rejected and that at the end of the program still 37.5% of proposed changes

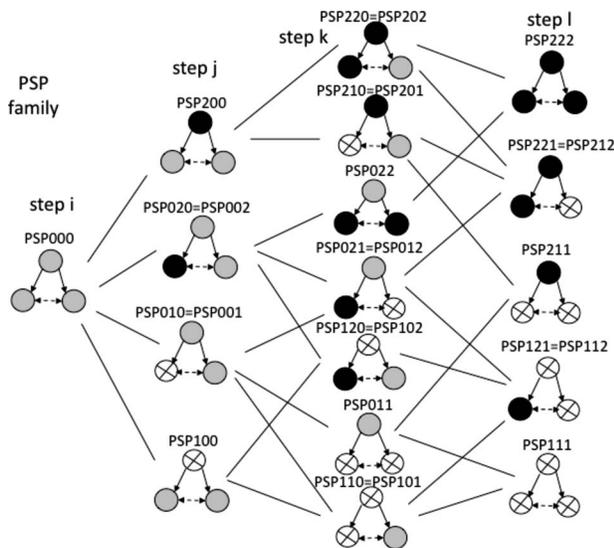


Fig. 14 Expanded PSP family of 3-motif change requests

Table 9 *CAI* and *CRI* calculations for the 11-CR network in Fig. 9

Area index	Number proposed	Number implemented	Number rejected	Total	<i>CAI</i>	<i>CRI</i>
3	1	0	1	2	0	0.5
10	0	3	1	4	0.75	0.25
14	0	0	1	1	0	1
17	0	4	0	4	1	0

Table 10 1-motif analysis results for entire data set

Entire data set	Proposed	Implemented	Rejected	Total
All CRs	15,611 (37.5%)	20,006 (48.2%)	5934 (14.3%)	41,551 (100%)
Isolates (0 deg nodes)	14,153 (54.2%)	9,672 (37.0%)	2288 (8.8%)	26,113 (100%)
Connected components	1,458 (9.5%)	10,328 (67.0%)	3642 (23.6%)	15,428 (100%)
87-CR (Fig. 8)	13 (15%)	53 (61%)	21 (24%)	87 (100%)

had not been decided either way. Distinguishing between isolates, i.e., change requests that are not involved in any parent-child or sibling relationships, and those in connected components (networks) helps shed light on these results. The majority of open change requests are isolates. According to interviews with project personnel, we found that many “nice to have,” nonessential design changes fell into this category. In connected components about two-thirds of proposed changes (67%) were implemented as they often required each other. In those change networks, about one in four changes (23.6%) were rejected. Empirical evidence tells us that many of the rejected changes were either not needed at all, or changes in other areas were substituted that tried to achieve the same or similar effect. The results for the 87-CR network shown in Fig. 8 are given in the last row of Table 10 and appear to mirror closely the average results found in the other change networks (connected components).

One of the original questions we set out to answer was whether some areas are more accepting of change than others. To shed light on this issue, we computed the *CAI* and *CRI* for all 46 areas of the system and all change requests in the data set. The results are tabulated in Table 11.

The most active areas in terms of actually implemented changes (more than 1000 implemented changes) are Areas 1, 3, 16, 19, and 32. Recall that Area 1 contained the system requirements and detailed design documentation. An interesting perspective can be gained by plotting *CRI* versus *CAI* for all areas (Fig. 15).

Areas that are in the lower right hand corner (below the diagonal) are classified as acceptors because they feature a larger fraction of accepted versus rejected changes. Area 38 at coordinate (1,0) is a perfect acceptor of change, meaning that all change requests issued against this particular subsystem were eventually implemented. Conversely systems above the diagonal are reflectors of change, since more changes issued against those areas are rejected rather than accepted. Those areas (such as 13, 39, 37, and 41) resist change to a stronger or lesser degree. It is very valuable for designers to understand why those areas resist change and to take this into account in future projects and product development efforts. Area 41 is particularly interesting since it is a perfect resistor of change. All ten change requests issued against this area over the course of the project (see Table 11) were rejected. It turns out that Area 41 contains a very complex and validated software algorithm that has been in use for many years and that only very few individuals understood in detail. Due to risk considerations, it was deemed more effective to leave Area 41 unaltered and to make changes in other areas instead.

A statistical analysis of the 2-motifs defined in Figs. 11 and 12, respectively, is summarized in Tables 12 and 13. The first row gives the results for all change requests in the data set. The second row gives the results for the 87-CR network only.

The results suggest that the most common parent-child pattern is P22, where both parent and child changes are implemented as planned (64.8%). If one of the two changes is rejected, it is roughly twice as likely to be the child (P21 19.3%) rather than the parent (P12 7.6%). The other parent-child 2-motifs are rather infrequent, particularly those where one of the two changes remains unresolved. The sibling 2-motifs exhibit similar statistics with pat-

Table 11 Change results and *CAI* and *CRI* by system area. The most heavily changed areas with more than 1000 implemented changes are in bold.

Area index	Number proposed	Number implemented	Number rejected	Total	<i>CAI</i>	<i>CRI</i>
1	102	4014	1079	5195	0.773	0.208
2	825	68	31	924	0.074	0.034
3	575	1893	621	3089	0.613	0.201
4	381	306	106	793	0.386	0.134
5	193	877	285	1355	0.647	0.210
6	127	545	226	898	0.607	0.252
7	140	853	237	1230	0.693	0.193
8	125	93	45	263	0.354	0.171
9	125	90	17	232	0.388	0.073
10	105	478	168	751	0.636	0.224
11	2201	544	212	2957	0.184	0.072
12	21	23	13	57	0.404	0.228
13	13	17	25	55	0.309	0.455
14	26	884	155	1065	0.830	0.146
15	16	101	40	157	0.643	0.255
16	436	1766	529	2731	0.647	0.194
17	471	149	27	647	0.230	0.042
18	1314	234	41	1589	0.147	0.026
19	186	2509	664	3359	0.747	0.198
20	24	135	28	187	0.722	0.150
21	8	30	15	53	0.566	0.283
22	94	51	41	186	0.274	0.220
23	43	306	80	429	0.713	0.186
24	226	172	21	419	0.411	0.050
25	93	594	151	838	0.709	0.180
26	54	7	5	66	0.106	0.076
27	145	0	3	148	0.0	0.020
28	252	40	11	303	0.132	0.036
29	86	21	7	114	0.184	0.061
30	83	38	18	139	0.273	0.129
31	90	165	36	291	0.567	0.124
32	88	1715	415	2218	0.773	0.187
33	646	135	17	798	0.169	0.021
34	0	0	0	0	0.6	0.6
35	1205	425	180	1810	0.235	0.099
36	0	34	10	44	0.773	0.227
37	0	3	4	7	0.429	0.571
38	0	7	0	7	1	0
39	2	7	9	18	0.389	0.5
40	1	3	1	5	0.600	0.200
41	0	0	10	10	0	1
42	1	258	79	338	0.763	0.234
43	0	50	22	72	0.694	0.306
44	4	22	12	38	0.579	0.316
45	0	104	57	161	0.646	0.354
46	3	240	107	350	0.686	0.306
Unknown (0)	5081	0	74	5155	0	0.014

terns S22 and S12 being the most frequent ones. It turns out that the 87-CR network mirrors the overall results quite closely and can therefore claim to represent the overall characteristics quite well.

So far we have not yet explicitly addressed the question as to what areas are prone to propagate or multiply changes toward other areas. Specifically we would like to be able to quantify (rather than guess or argue qualitatively) which areas acted as multipliers, carriers, or absorbers [2] of change. To unambiguously define change propagation, we state the following.

Change propagation occurs when a proposed (parent) change in one area leads to an *implemented* (child) change in another area, regardless whether the parent change is ultimately implemented or not. Sibling changes do not imply true change propagation as they only indicate that two changes are related and not that one spawned the other. With this strict definition in mind, change propagation only occurs when the following parent-child 2-motifs

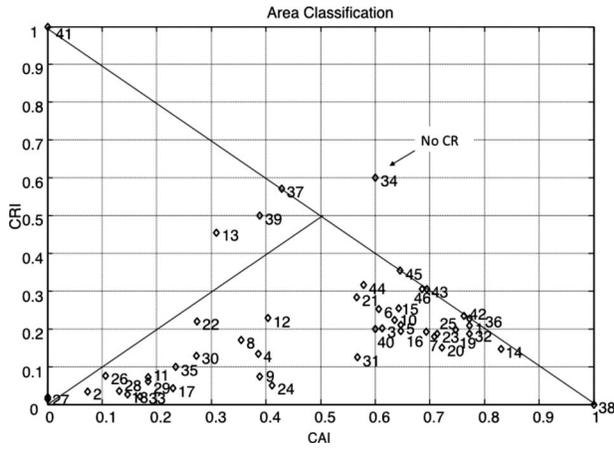


Fig. 15 Area classification for *CRI* and *CAI* (entire data set)

are present: P02, P12, and P22 (see Fig. 11). In Sec. 9, we use this definition to compute a *CPI* for each area.

9 CPI

A *CPI* has been defined before [8,13], but its definition and exact calculation have remained somewhat ambiguous. The following explanation clarifies the calculation of *CPI* for each area. Let the sum of changes from all N areas, including self-changes affecting area i be $C_{in}(i)$, the in-degree, and the sum of change values from area i to all N other areas be $C_{out}(i)$. Only the 2-motifs P02, P12, and P22 are used to determine C_{in} and C_{out} . The $CPI(i)$ for area i is then simply

$$CPI(i) = \frac{C_{out}(i) - C_{in}(i)}{C_{out}(i) + C_{in}(i)} \quad (3)$$

and is a number between -1 and $+1$. Table 14 shows a sample calculation for the 11-CR network in Fig. 9.

A value of $+1$ indicates a perfect *multiplier*, i.e., a component or area that only originates changes and has no incoming changes, such as Area 14. A value of -1 indicates a perfect *absorber*, i.e., a component or area that only receives changes but generates none (or passes none along), such as Area 17. Most areas are expected to have a mix of incoming and outgoing changes such as Area 10, which appears to be a weak multiplier. Areas with a *CPI* of zero (or close to zero) can be classified as *carriers*.

The $CPI(i)$ can be computed by area for each of the networks identified in Table 5 or for all changes in the project. When all data are used, a characterization of an area as a whole is obtained. However, caution is advised, since some areas could act as multipliers for some specific types of changes, e.g., when trying to increase the response rate of the system, while acting as carriers or absorbers for other types of changes such as decreasing the rate of false positive alarms that the system generates. Table 15 shows the *CPI* calculation for the entire data set, thus quantifying the general propensity of an area to act as a propagator of change in the system.

Table 13 Sibling 2-motif analysis for all change requests

	Sibling motifs						Total
	S00	S01	S02	S11	S12	S22	
All conn	278	574	2115	2056	7467	12075	24565
comp	(1.1%)	(2.3%)	(8.6%)	(8.4%)	(30.4%)	(49.2%)	(100%)
8-CR	8	4	16	14	48	68	158
(Fig. 8)	(5.1%)	(2.5%)	(10.1%)	(8.9%)	(30.4%)	(43%)	(100%)

With the *CPI* computed, the system areas can be shown on the absorber-to-multiplier spectrum between -1 and $+1$ as in Fig. 16. The upper view shows the entire *CPI* spectrum between -1 and $+1$, while the lower view zooms in on the center $[-0.4, 0.4]$ region. Three areas appear as perfect multipliers: 27, 41, and 45; however, each of these areas has less than five outgoing changes recorded. If we consider systems with a $CPI > 0.3$ (an arbitrary delimiter) as strong multipliers, then areas 1, 13, 22, and 46 also fall into this category. Area 1, in particular, contains the requirement documentation and it is intuitive that this area may act as a multiplier. The perfect absorbers 26, 36, 39, and 44 have less than five changes implemented, while Areas 9, 24, and 31 appear to be strong absorbers receiving significantly more changes than they generate. Another example of a strong absorber is Area 3, the graphical user interface (GUI) of the system. Weak absorbers, such as Areas 16 (hardware performance evaluation) and 5 (core data processing logic), have a *CPI* between -0.1 and -0.3 , while carriers are defined as having a *CPI* between -0.1 and 0.1 . Examples of carriers are Areas 25 (hardware functional evaluation) and 32 (system evaluation tools).

Overall we counted 7 strong multipliers, 3 weak multipliers, 6 carriers, 13 weak absorbers, and 13 strong absorbers. Thus, there appears to be a preponderance of change absorbers in the system. The significance of the multipliers is that they are good candidates for focused change management and oversight by integrated product teams (IPTs). Generally, multipliers are also considered to be good candidates for embedding flexibility [13]. Note, however, that to gain a more complete picture as to where flexibility, i.e., the ease of making changes, is potentially most valuable, one would also have to consider the number of changes as well as the average change magnitude by area, since change magnitude correlates directly with change effort [18–21].

9.1 Visualization. In order to experiment with change visualization methods, the data were also used to form an a posteriori change model in the CPM tool developed by Keller et al. [11]. The likelihood of propagation occurring from one area to another was computed as a simple proportion by counting the number of instances where changes were linked from area m to area n , then dividing by the total number of changes in area m . Figure 17 is a propagation tree generated by the CPM tool showing possible propagation paths resulting from a change in Area 1, the strong multiplier identified earlier. The distance (number of rings) of each node from node 1 is proportional to the probability of a change propagating from Area 1 to the area represented by the node. Figure 17 is a distorted (fisheye) view with the focus on

Table 12 Parent-child 2-motif analysis for all change requests

	PC motifs									Total
	P00	P01	P10	P11	P02	P20	P21	P12	P22	
All	9	46	6	275	98	27	1075	426	3619	5581
comp	(0.2%)	(0.8%)	(0.1%)	(4.9%)	(1.8%)	(0.5%)	(19.3%)	(7.6%)	(64.8%)	(100%)
87-CR	0	2	0	1	3	0	10	2	23	41
(Fig. 8)	(0%)	(4.9%)	(0%)	(2.4%)	(7.3%)	(0%)	(24.4%)	(4.9%)	(56%)	(100%)

Table 14 Sample calculation for *CPI* for the 11-CR network in Fig. 9

Area	Out-degree	In-degree	<i>CPI</i>
3	0	0	N/A
10	3	2	0.2
14	1	0	1
17	0	2	-1

Area 32.

We found that the CPM tool was effective but that change propagation is not simple to characterize for a system of this size. In particular, the 46 areas chosen for modeling are more akin to subsystems rather than individual (mechanical) components, allowing the possibility of multiple distinct changes being necessary to change a characteristic function of the system.

As a result of the combinations of data, which were available for extraction, and the different methods for displaying and inter-

Table 15 *CPI* calculation for entire data set (using P02, P12, and P22 motifs)

Area	Out-degree	In-degree	<i>CPI</i>
1	1588	425	0.577
2	9	13	-0.182
3	270	521	-0.317
4	74	73	0.007
5	223	303	-0.152
6	96	91	0.027
7	171	237	-0.162
8	17	12	0.172
9	11	37	-0.542
10	94	136	-0.183
11	134	107	0.112
12	9	12	-0.143
13	2	1	0.333
14	142	220	-0.215
15	16	33	-0.347
16	345	464	-0.147
17	17	35	-0.346
18	28	52	-0.300
19	354	715	-0.338
20	13	15	-0.071
21	3	6	-0.333
22	31	15	0.348
23	18	27	-0.200
24	17	58	-0.547
25	89	105	-0.082
26	0	1	-1.00
27	1	0	1.00
28	5	8	-0.231
29	5	3	0.250
30	8	16	-0.333
31	15	46	-0.508
32	253	283	-0.056
33	8	12	-0.200
34	0	0	N/A
35	27	23	0.080
36	0	3	-1.00
37	0	0	N/A
38	0	0	N/A
39	0	1	-1.00
40	0	0	N/A
41	1	0	1.00
42	15	21	-0.167
43	0	0	N/A
44	0	3	-1.00
45	3	0	1.00
46	23	10	0.394
Unknown (0)	11	0	1.00

preting that data (including use of network depiction, DSMs, and the Cambridge EDC's CPM tool), several interesting patterns emerged for consideration as described above and in more detail in Ref. [8].

10 Change Patterns Over Time

The data also showed the rate at which change requests were being generated. Figure 18 shows the number of new change requests written per month over the duration of the project.

Features of particular interest include not only the overall curve, which is very similar in shape to that expected of staffing levels for a large project, but the ripples atop the curve (shown as an oscillating solid line). Coinciding with the creation of new integrated product teams (IPTs) in month 40, we see a marked change in the rate of change request generation. The most salient feature of an IPT is the incorporation of a wide range of disciplines into regular evaluation of a project. Here, around month 40 the project shifted from just those who were working in their own functional areas of development being able to see and question their components to simultaneous evaluation by those concerned with test and system level integration. It is unsurprising that the number of change requests would climb significantly, with the introduction of fresh eyes and different assumptions to the mix. The overall pattern seen here of accelerated discovery of problems (and thus rework) corresponds to a typical observation from modeling projects with System Dynamics [1].

An interesting observation is that the pattern observed in this research is qualitatively similar to the "ripple" pattern predicted by Eckert et al. [2], reproduced in Fig. 19 below, thus providing the first quantitative evidence for this particular change pattern. Unlike the "early" ripple predicted, however, we observed a "late ripple" with the peak of change activity occurring late in the program due to extensive software system integration and testing.

It stands to reason that a large and long running program like this one might progress through cyclical phases during which the focus shifts from implementation to testing to fixing and on back (see superimposed "oscillations" in Fig. 18). The effects in terms of rate of change request generation would be amplified atop a curve delineated by overall project staffing (more people looking for bugs will probably find a greater number). Potential future work would be to recreate this change pattern using a system dynamics-based rework model.

11 Summary and Conclusions

This paper focused on mining and analyzing a large data set containing over 41,500 change requests generated during the design of a complex sensor system. The data set describes the design evolution of the system over a period of approximately 8 years. The purpose of the research was to better understand the nature of change and change propagation, and to draw out applicable solutions for future development programs of similar scale. Design documentation was used to create a DSM representing the intended structure of the program, and then the data were analyzed to yield a change DSM, describing the actual change structure of the program. The main contributions of the paper are the formalization of the idea of change motifs (one-, two-, and three-node motifs) as a means of systematically analyzing change networks as well as the definition of three indices to quantify each area in terms of its propensity for accepting (*CAI*), reflecting (*CRI*) or propagating changes (*CPI*).

Conclusions regarding the combined information were drawn, including the following.

- Change propagation was a consistent occurrence in the development of this large technical system.
- Change activity was not uniformly distributed throughout the system but more concentrated in some areas. The areas that featured more than 1000 implemented changes were 1, 3, 16, 19, and 32.

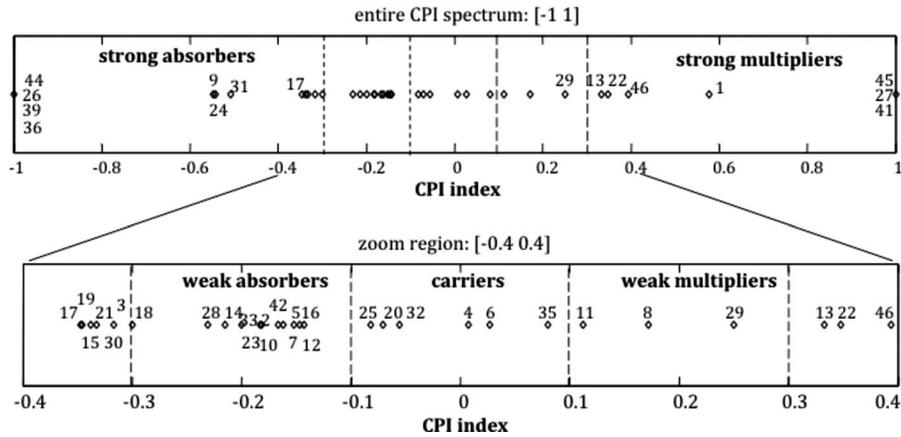


Fig. 16 CPI spectrum between -1 and 1 with areas mapped

- (c) Change requests were generated even between areas that did appear not to be directly connected in the system DSM shown in Fig. 6. While counterintuitive when thinking about purely mechanical systems, it appears that changes can propagate between not directly connected areas in systems that have both high electro-mechanical and software complexity.
- (d) A small number of areas that involved more than ten implemented changes acted as strong multipliers ($CPI > 0.3$). These areas were 1, 13, 22, and 46 and contained either top-level requirements and program documentation or were located at the intersection of major functional areas. These strong multiplier areas are potentially good

candidates for more focused change management, IPTs, and embedding flexibility to reduce the amount of change propagation to other areas.

The results of the change request structure were also analyzed using network graphs and one-, two- and three-change motifs. A change network can be obtained by mapping all parent-child and sibling change request ID numbers. The largest change networks that were found had 2566, 445, 170, 87, and 64 related changes, which was a surprise to us. It was found that these large change networks did not emerge from a single parent change but by coalescence of initially disconnected changes that nucleated across the system at different times.

While the research presented here focused on a posteriori analysis of change activity in a completed project, a number of valuable insights were gained that could be applied in a forward-looking manner toward the next project. Examples of such insights include the following.

1. An understanding of which areas consistently resisted change.
2. The fraction of change requests that was generally rejected. This is a measure of wasted effort since every change request that was ultimately rejected had to be evaluated in the first place.
3. Change patterns that could be expected to occur with the greatest likelihood. This could have profound implications for the choice of better system architectures and design process models.

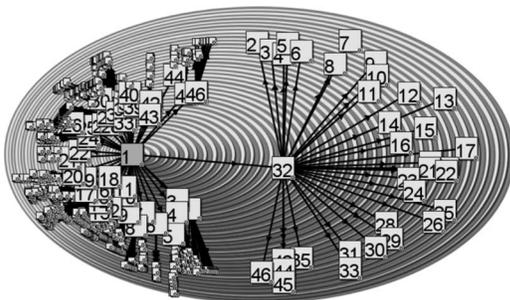


Fig. 17 Change propagation tree focused on changes propagating from Area 1 to Area 32 (CPM tool)

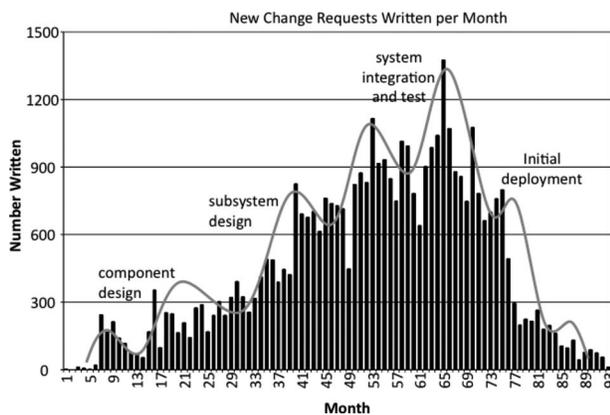


Fig. 18 Change requests written over time (number written per month)

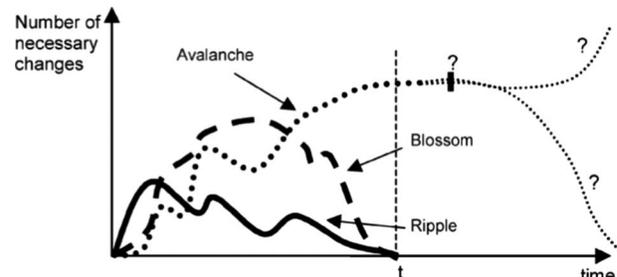


Fig. 19 Change patterns predicted in Eckert et al. [2]

tion is absolutely critical. If that level of access and cooperation between academia and industry can be attained, the following are promising areas for future work.

- a. *Change prediction*: Evaluate change components (networks) and paths in detail versus those predicted by current tools. How good are our predictions regarding propagation paths versus actual changes? How good are our predictions regarding actual versus planned effort? How can they be improved? How can change propagation patterns observed on past projects be leveraged for design decisions (e.g., drawing module boundaries, actively embedding change absorbers, and decomposing multipliers) in new projects?
- b. *Data processing*: Standardize methods for refining data, tracing large change networks in greater depth-attempt to reconstruct logic, particularly where proposed changes were rejected or were superseded by other changes. Gather rationale underlying all 2- and especially 3-motif patterns.
- c. *Staffing*: Analyze effects of staffing on changes and components. What differences occur in change propagation patterns based on which personnel work on the changes? Can change-related personnel information be used for performance and status assessments in technical organizations? Are there relationships of interest between the social network, system network, and change management network in the system?
- d. *Contractual*: Can change propagation information be used to write better prime and subcontracts such that contractual changes and amendments can be incorporated more easily in those projects involving multiple firms?
- e. *Statistical*: Are there critical numbers for change propagation? Limits on the number of propagation steps? For example, in the 11-CR network (Fig. 9), the longest chain of contiguous parent-child changes had a length of 2. Previous research has suggested that change propagation beyond four steps is very rare, but we have not yet investigated this assertion. If a change network grows, are additional changes more or less likely? As change networks grow, does the average change magnitude and likelihood of change implementation change? It would be very beneficial to include change magnitudes (Table 2) into propagation paths and impact analysis.

Ideally, future change propagation research will include partnering with multiple firms to set up a standardized information collecting system and track all of the data noted here (and more), as well as making evaluation of the data part of regular post-project assessments. Hopefully these avenues and more will be explored, leading to a better understanding of change propagation in the quest to design future complex systems and products with less uncertainty regarding the amount of expected engineering change and rework.

Acknowledgment

We acknowledge the financial support and data access provided by Raytheon Integrated Defense Systems.

Nomenclature

- $C_{in}(i)$ = total number of inbound changes implemented in area i
- $C_{out}(i)$ = total number of outbound changes from area i implemented in other areas
- N = number of system areas (=“subsystems”)
- P = parent
- p_{ij} = probability of a change propagating from area i to j
- S = sibling

References

- [1] Sterman, J. D., 2000, *Business Dynamics: Systems Thinking & Modeling for a Complex World*, McGraw-Hill/Irwin, New York.
- [2] Eckert, C., Clarkson, P., and Zanker, W., 2004, “Change and Customization in Complex Engineering Domains,” *Res. Eng. Des.*, **15**, pp. 1–21.
- [3] Brown, J., 2006, *Managing Product Relationships: Enabling Iteration and Innovation in Design*, Aberdeen Group, Boston.
- [4] Wright, I. C., 1997, “A Review of Research Into Engineering Change Management: Implications for Product Design,” *Des. Stud.*, **18**, pp. 33–42.
- [5] Pikosz, P., and Malmqvist, J., 1998, “A Comparative Study of Engineering Change Management in Three Swedish Engineering Companies,” *Proceedings of the DETC98 ASME Design Engineering Technical Conference*, pp. 78–85.
- [6] Martin, M. V., and Ishii, K., 2002, “Design for Variety: Developing Standardized and Modularized Product Platform Architectures,” *Res. Eng. Des.*, **13**, pp. 213–235.
- [7] 2002, United States, DoD Directive 5015.2, Department of Defense Records Management Program, Jun. 19.
- [8] Giffin, M., 2007, “Change Propagation in Large Technical Systems,” SM thesis, System Design and Management Program, MIT, Cambridge, MA.
- [9] Huang, G. Q., and Mak, K. L., 1999, “Current Practices of Engineering Change Management in UK Manufacturing Industries,” *Int. J. Operat. Product. Manage.*, **19**(1), pp. 21–37.
- [10] Clarkson, J., Simons, C., and Eckert, C., 2004, “Predicting Change Propagation in Complex Design,” *ASME J. Mech. Des.*, **126**(5), pp. 788–797.
- [11] Keller, R., Eger, T., Eckert, C., and Clarkson, J., 2005, “Visualising Change Propagation,” *15th International Conference on Engineering Design*, pp. 62–63.
- [12] 2006, *Product Platform and Product Family Design: Methods and Applications*, T. W. Simpson, Z. Siddique, and J. Jiao, eds., Springer, New York.
- [13] Suh, E. S., de Weck, O. L., and Chang, D., 2007, “Flexible Product Platforms: Framework and Case Study,” *Res. Eng. Des.*, **18**(2), pp. 67–89.
- [14] Terwiesch, C., and Loch, C. H., 1999, “Managing the Process of Engineering Change Orders: The Case of the Climate Control System in Automobile Development,” *J. Prod. Innovation Manage.*, **16**, pp. 160–172.
- [15] Cohen, T., and Fulton, R. E., 1998, “A Data Approach to Tracking and Evaluating Engineering Changes,” *Proceedings of the DETC98 ASME Design Engineering Technical Conference*, Paper No. DETC98/EIM5682.
- [16] Eppinger, S., Whitney, D., Smith, R. P., and Gebala, D. A., 1994, “A Model-Based Method for Organizing Tasks in Product Development,” *Res. Eng. Des.*, **6**, pp. 1–13.
- [17] Smaling, R., and de Weck, O. L., 2007, “Assessing Risks and Opportunities of Technology Infusion in System Design,” *J. Syst. Eng.*, **10**(1), pp. 1–25.
- [18] Earl, C., Eckert, C., and Clarkson, J., 2005, “Design Change and Complexity,” *Second Workshop on Complexity in Design and Engineering*.
- [19] Jarratt, T., Eckert, C., and Clarkson, J., 2004, “Development of a Product Model to Support Engineering Change Management,” *Proceedings of the TMCE, Fifth International Symposium on Tools and Methods of Competitive Engineering*, Lausanne, Switzerland, April 13–17.
- [20] Jarratt, T., Eckert, C., and Clarkson, J., 2005, “Pitfalls of Engineering Change: Change Practice During Complex Product Design,” *Advances in Design*, pp. 413–424.
- [21] Keller, R., Eckert, C., and Clarkson, J., 2005, “Viewpoints and Views in Engineering Change Management,” *GIST Technical Report No. 1*.