INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

CHANGE PROPAGATION AT THE INTERFACE OF SYSTEM AND EMBEDDED SOFTWARE DESIGN: CHARACTERISING IMPACT ANALYSIS TASKS AND TECHNIQUES

MS Kilpinen[1], CM Eckert[1] and PJ Clarkson[1]
[1]University of Cambridge, Engineering Design Centre

ABSTRACT
Embedded software is specifically designed for and integrated into the control system of a product. This software provides for flexibility in a product design by allowing for the accommodation of changes originating from mechanical or electrical designs. Within mechatronic products, embedded software can modify the functionality of a product without necessarily requiring the redesign of physical components. System designers coordinate the changes handled by embedded software for the mechanical and hardware designs. The perception that the software can *easily* be changed is often not the case. The synchronisation of cross-disciplinary design modifications with the embedded software design can be particularly complex, and the initiating changes often cause additional, unexpected modifications. Understanding the complete impact of a modification on the embedded software design is vital to managing this propagation of changes; unanticipated modifications create rework and unexpected costs during development.
System and software designers perform change *impact analysis* during the design process to assess the scope or consequences of implementing a change in order to determine if the change should be carried out and to mitigate the occurrence of emergent changes within the embedded software design. The research presented investigates the range of impact analysis techniques available within system and software design and characterises the rigour of and influences on these tasks when implemented in practice, as identified through two empirical studies. Means to improve the quality of impact analysis are suggested through specific scenarios elicited from industry.

Keywords: Change impact analysis, change propagation, systems engineering, embedded software

1 INTRODUCTION
Designers increasingly incorporate embedded software within traditional mechanical and electronic hardware designs to support or enhance a product's functionality. Within such a cross-disciplinary environment, system designers are specifically designated in some cases to coordinate the subsystem designs of mechatronic products by developing the product requirements and allocating these requirements to the mechanical, hardware, and software engineers for detail design [1]. A mechanical, electrical, or software engineer may perform the integration of the subsystem designs. As the subsystem engineers concurrently generate the detail designs, changes due to new product requirements, errors, or design improvements may cause system designers to modify the requirements or distribution of requirements to the subsystems. Modifications can occur throughout the design process, potentially leading to late product delivery and high development costs.
Synchronising the software design with the other subsystems can be particularly challenging for system designers. As changes occur during the product development, system designers may often attempt to contain the design modifications within the software design, leaving the mechanical and hardware designs untouched; the adaptable nature of software code is relied upon to absorb or accommodate the changes [2]. In some cases, late changes to the software design may be required to adapt the functionality of physical components since the hardware has already been manufactured. However, the complex interdependencies of the software and the other subsystem designs make the full impact of a change difficult to determine by system, mechanical, electrical, or software designers.

In turn, a change to the software design can cause additional, unexpected modifications within the software design as well as produce a ripple of further changes to the high-level software requirements or functionalities specified by the system designers. These emergent changes often make the initial software modification much more complex than predicted and can incur a significant increase in expected development costs. Changing the software may not prove cheaper than modifying the mechanical or electrical designs in the end.

In order to initially identify these emergent or knock-on effects of a change, system and software engineers may perform different styles of change *impact analysis* (IA). IA refers to the investigation into the modification and intends to estimate the consequences of the change or determine what needs to be modified in order to accomplish a design change [3]. This system-software analysis allows for the prediction of how the change affects the product design, including the system requirements, software specifications, and software design or code, and has similar intentions to change prediction in mechanical design [4]. While some changes may undergo very thorough IA by system and software designers, other modifications may be minimally analysed, if at all [2]. Through two empirical studies, the research presented characterises the IA performed in practice in terms of rigour and the influences designers cite as triggering their selection of IA technique. Analysis of specific scenarios of IA implemented by designers suggests how the quality of the IA executed can be improved, leading to strategy for reducing unanticipated change propagation.

2 METHOD

This research conducted a systematic literature review in the systems, mechatronic, mechanical, and software engineering disciplines on the application of IA within change processes. The focus was to determine how IA is prescribed within each discipline and how IA is applied across the software design interface. The literature found to date (primarily from systems and software engineering) reveals a range of possible IA and also suggests that designers may not perform IA as prescribed in practice (as discussed in section 3.1). However, the impact of implementing less rigorous forms of IA and guidance on how to improve the IA performed are not thoroughly identified in this body of literature.

In order to investigate the implementation of IA in practice, two empirical studies at large engineering firms were conducted in the aerospace and telecommunication industry sectors. In the aerospace company, embedded software supports the functionality of a primarily mechanical product. This firm advocates using a systems engineering design approach in which system designers develop the high-level software requirements. In contrast, the telecom company develops products integrating hardware and software, and the embedded software delivers most of the functionality of these products. Agile software design processes are implemented the latter company. The empirical studies provide insights into the IA practised within systems and software engineering change processes and allow for the development of IA characterisations (presented in section 4). In order to clarify the practice of IA, the characterisations aim to describe rigour of and the difficulties in implementing various styles of IA. These characterisations then allow for a systematic analysis of IA improvement scenarios.

2.1 Empirical studies

The empirical studies collected data through a grounded-theory informed approach [5]. Semi-structured interviews, numerous informal discussions, daily observation, and documentation review with key stakeholders, including (1) system designers, (2) embedded software designers, (3) design process engineers, and (4) managers, obtained a holistic perspective on the interaction of the systems and software domains. Interviews were transcribed and analysed by identifying and revising overarching concepts and themes. The stabilisation of these themes in the subsequent interviews suggests a common understanding had been reached.

2.1.1 Phase 1: Exploratory study

An initial 7-week study on-site at the aerospace company aimed to gain insight into the change processes at the interface between systems and embedded software engineering. The exploration allowed for the understanding of the context of the change practices. The study enquired into the following three areas:

1. The change and requirement management processes in practice,
2. The iterative design and requirement management processes prescribed by the company, and

3. The change IA performed by the systems and software designers.

Twenty-five interviews, lasting between one and two hours each, elicited a variety of IA implemented and even no IA within a change process. In addition, the interviewees often mentioned the influences and difficulties in implementing IA.

In the analysis of this initial study, it was noted that designers perceived many difficulties in implementing some forms of IA. Most interviewees would cite barriers to performing a particular prescribed-form of IA, but did not discuss the selection process of the IA executed. This research develops characterisations to summarise the different types of IA performed in practice and the influences to the selection of these IA tasks captured in the empirical studies. During the first phase of the empirical studies, initial models of these characterisations were created. These models were subsequently refined to create the characterisations (presented in section 4) after the second phase of the empirical studies.

2.1.2 Phase 2: Impact analysis study

The second phase consisted of a follow-up study at the aerospace company, including 15 additional interviews approximately a year after the exploratory study. Ten of these 15 interviewees had been interviewed in phase 1, and the remaining interviewees were recruited through contacts suggested during the second phase of interviews. The interviews focused specifically on the nature and selection process of the IA implemented and aimed to:

1. Identify the range of IA implemented in practice,
2. Determine other IA prescribed or possible within the company, which might not be performed,
3. Elicit the influencing factors in the IA selection process or barriers to not implementing IA, and
4. Quantify the inputs and outputs of the IA implemented.

These interviews systematically discussed IA practiced through specific change examples. Data regarding the quality of information, time, resources available for the IA was collected as well as the perceived quality of the IA prediction for each change case. The interviews specifically elicited a wider range of influences on the IA selection process from the first empirical study phase and also illuminated several areas for improvement and expansion of the characterisation.

In addition, the second phase benefited from 9 discussions during a 3-month period and numerous informal discussions during a 3-day design workshop at the telecommunications company. These discussions centred on the first three elements of the phase 2 interviews at the aerospace company, as described in the bullet points above. This empirical study contributes a perspective on IA based on an agile software design process and contextualises the results obtained from the aerospace company.

3 IMPACT ANALYSIS

Systems and software engineering literature suggests several basic means or techniques to perform IA at the systems-software interface, as identified within the empirical studies. Within the empirical studies, the system designers negotiated the desired functionality of the software with the mechanical and electrical engineers involved in the product development. Software designers did not directly interact with the mechanical or electrical engineers typically; all modifications from these stakeholders would first be analysed by the system designers. The system and software designers observed primarily interfaced through requirement documentation. Systems engineers developed the high-level software requirements, which were passed to the software designers for detailed specification and design. In some cases, the requirements were visually modelled or animated in order to clarify text.

Research found to date on IA primarily focuses on developing methods or tools to investigate the spread of changes across a product in novel ways while other literature found in system and software engineering alludes to the difficulties in implementing IA in practice. The following discussion sets the context of IA within systems and software engineering change processes, as observed in the empirical studies. This research then proposes a perspective on the disparity between the IA prescribed with regards to the IA actually applied in industry.

3.1 Analysis at the system and embedded software interface

Within systems and software engineering literature, change processes frequently prescribe the maintenance of design traceability. Traceability refers to the mapping of product requirements to their respective detail specifications and designs within the subsystems (i.e. system requirements are linked to more detailed software specification, which also trace to software design artefacts, such as models,

documentation, or code files). The literature suggests that maintaining these relationships allows for completeness in the product design documentation throughout the design process [1]. In turn, these traceability relationships allow the engineers to perform *traceability IA* across the system-software interface [6], as observed in the empirical studies. Given a change to a system requirement or software design, the existing traceability relationships extending from the effected element can be identified. These traceability relationships indicate the possible system requirements, software specifications, or even specific software artefacts to which a change may propagate. Several commercial computer tools allow for the automation of this style of IA by structuring a database of traceability relationships among requirements and design artefacts. Other forms of design or change documentation not organised in such a database can also be used for impact analysis by manually searching references between and within documents. However, this search process is difficult to do exhaustively.

Software engineering literature also suggests another style to investigate the impact of changes through *dependency IA* [3]. By identifying the linkages between variables, logic, modules, etc. within software models and code, a detailed analysis of the impact of a change can be performed. For example, software requirements or software design models [7] or actual code may be searched for dependencies between modules or variables [3]. This search may be automated or performed manually. With the search results, designers can determine how a change explicitly affects the software design. However, this dependency analysis tends to focus on specific low-level changes rather than functional modifications, which traceability IA potentially could identify. For instance, dependency IA may not be able to diagnose the timing and synchronisation of the calling of software functions, requiring a high-level analysis of the distribution of software requirements and timing constraints. Dependency IA may be used in conjunction with traceability IA for more detailed analysis of software designs.

Agile software design processes propose an alternative means of IA. As opposed to systems and some software engineering literature, these design methods focus on working software rather than documentation [8]. Agile processes tend to rely on quickly responding to changes through collaboration with the customer and within the design team to perform IA. Given this perspective, an argument against using traceability IA can be made. Ambler contends that traceability IA should not be used since the cost of maintaining traceability databases is high and suggests that designers familiar with the system should simply use their experience and engineering judgement to estimate the scope of a modification (termed *experiential IA* by this research) [9]. Engineers may perform experiential IA by simply thinking about the implications of a change or discussing proposed modification within the design team. Experiential IA can identify tacit dependencies and, therefore, mechanisms for change propagation through expert knowledge, typically not captured by models or databases. However, this form of IA may be unsystematic by nature, potentially neglecting means of change propagation. Nevertheless, agile design processes do not exclude the implementation of traceability and dependency IA, and vice versa, as observed in the telecom company.

Within the traceability, dependency, and experiential forms of IA several more specific implementations of IA exist. Figure 1 highlights some of these instances of IA, but should not be considered exhaustive. Traceability IA can be automatically performed through relationships between requirements and design artefacts or manually through lower-level of design documentation; dependency IA can similarly be implemented on design models or the actual elements or code of the design; and, experiential IA can occur by applying individual engineering judgement or through discussions within a variety of contexts.

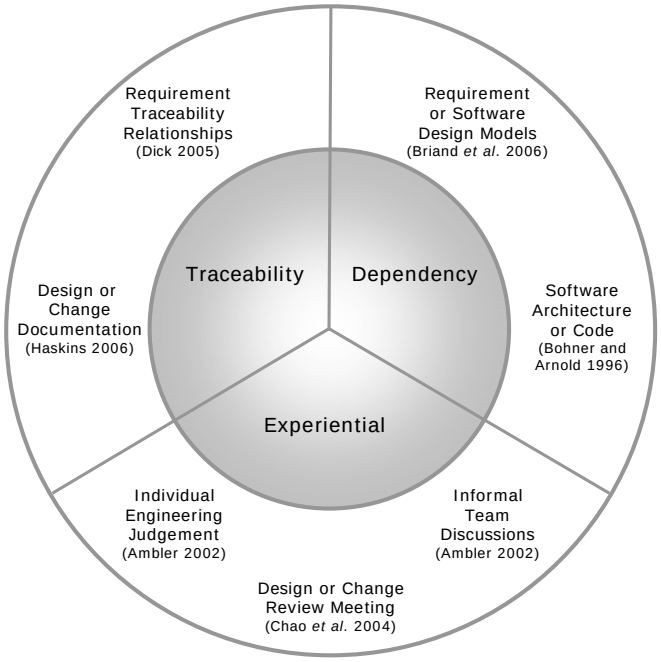

Figure 1. The range of IA techniques

3.2 Classifying impact analysis tasks and techniques

As previously mentioned, systems and software engineering research cites difficulties in adapting these forms of IA for industry practice. For instance, literature indicates that modifications are identified through experience and often are implemented in an *ad hoc* manner [2, 10]. Despite the fact that systems and software engineering literature frequently recommends performing traceability IA, Graaf *et al.* discuss the lack of maintaining accurate requirement documentation and traceability throughout the product development process as a root cause to this lack of IA implementation [11]. In turn, Dick suggests that traceability IA must be used in conjunction with engineering judgement to ascertain the actual impact of a change as opposed to the possible propagation of changes given by the traceability analysis [6]. Thus, traceability IA is not without limitations. Similarly, Lindvall and Sandahl indicate in their studies that experiential IA often produces incomplete results of the impact of a software design modification [12]. Literature on specific forms of dependency IA tends to mention more technical difficulties in implementing these approaches to IA with the stipulation that the results are limited to the information captured within the model or code. Furthermore, other influences, such as the time and effort required to perform traceability, dependency, or experiential IA, can limit the execution of the analysis due to project planning and scheduling constraints.

This set of literature illustrates a theme; the idealised forms of IA may not be exactly implemented as prescribed in practice. This research project distinguishes this disparity within IA by taking *IA techniques* to include the theoretical or possible methods or approaches to estimating the scope a change (e.g. traceability IA) and *IA tasks* as the actual implementation of IA technique within a particular change context (e.g. without complete traceability relationships). Thus, an IA task may not provide results at as high *quality* (i.e. with completeness, correctness, and clarity) compared to what an IA technique theoretically produces due to external influences.

The lack of a procedure to systematically capture and update traceability relationships can lead to incompleteness of the captured relationships and, in turn, reduce the quality of IA results by limiting the implementation of the traceability IA technique. In contrast, a lack of information, resources, or time to perform the analysis influences the specific traceability IA task applied. Moreover, the availability of information, resources, and time can also affect IA results despite the choice of IA technique (i.e. traceability, dependency, or experiential). For instance, inaccurate information about what needs to be changed will cause degradation in IA results to some extent from both a highly-detailed dependency IA or a less-detailed experiential IA. Thus, there are *influences* that either can affect (1) an IA technique or (2) an IA task (discussed in detail in section 4.2).

This research proposes that increasing the quality of IA results reduces the likelihood of emergent, knock-on changes occurring during a change process. As a means for improving the design process,

the range of influences on the differences between IA tasks and techniques is first characterised through the observations within the empirical studies. By first baselining current industry practice, a strategy for addressing these influences is described within the scope of these empirical studies.

4 CHARACTERISING THE DISPARITY OF IA TASKS AND TECHNIQUES

Within the empirical studies, designers discussed the implementation of IA techniques, the quality of IA results, and the influences they perceive to produce a reduction of the quality of results. These studies informed the development of two characterisations to describe the differences between IA tasks and techniques and the influences causing this difference.

4.1 Defining the theoretical rigour of IA techniques

In order to characterise the disparity between the actual implementation of IA tasks in practice and the generic techniques of performing IA, this research introduces the concept of IA *rigour*. While the term rigour can be interpreted in many different ways and contexts, this research takes the rigour of IA to correspond to the risk of unexpected change propagation; the size of the search space of possible change propagation paths primarily determines the risk of change propagation in theory. Missing a means of change propagation during IA increases the risk of emergent changes. More rigorous IA is expected to reduce the tendency of emergent changes while less rigorous IA may lead to unanticipated knock-on modifications.

Figure 2 suggests the range of rigour for instances of IA techniques. This characterisation does not aim to classify all IA techniques possible and only suggests a relative rating of common IA techniques based on the theoretical perspective on rigour taken. Highly rigorous IA is expected to systematically search all possible means of change propagation, virtually eliminating the potential for unanticipated, knock-on changes. Less rigorous IA may perform a more *ad hoc* or unsystematic search process and have a risk of requiring emergent modification.

IA through requirement traceability relationships is shown as the most rigorous technique in Figure 2. This method enables the systematic (and often automated) search of the impact of a change from high-level product functionalities to software detail designs. In theory, similar results can be obtained through manual searches through documentation and references. Requirement traceability relationships may capture additional design dependencies other than referenced explicitly in the documentation. Other methods may not include such a comprehensive search of the implications of modifications on broad functionalities.

Dependency IA may provide a less thorough search of possible paths of change propagation and, thus, be less rigorous than traceability IA. System requirement or software design models or actual software design artefacts (e.g. architecture or code) may not necessarily connect the detail design with overarching system functionalities, resulting in a limited search space compared to traceability IA. System and software design models or code can often abstract away from higher-level requirements of the product design, and the interdependencies due to mechanical or physical constraints may be lost in the decomposition of the product into different software modules or functionalities. More detailed software design artefacts may contain even less information to determine the implications of a change on higher-level system architecture than system or software design models. Thus, Figure 2 places the IA technique by searching models at a higher level of rigour than by analysing software design code.

Experiential IA is classified at the lower-end of the rigour scale. While expert judgement can provide the best form of IA in a situation due to tacit knowledge and latent dependencies not captured by models, the method for searching all possible paths of change propagation is often more *ad hoc*. Thus, emergent changes can occur using this IA technique, and the rigour of experiential IA is lower than traceability and dependency IA. However, by increasing the number of experts or individuals performing the IA, the IA rigour may increase given that the search space theoretically increases. The systematic discussion of a design with a variety of relevant stakeholders, such as in a review meeting, may further increase the IA rigour.

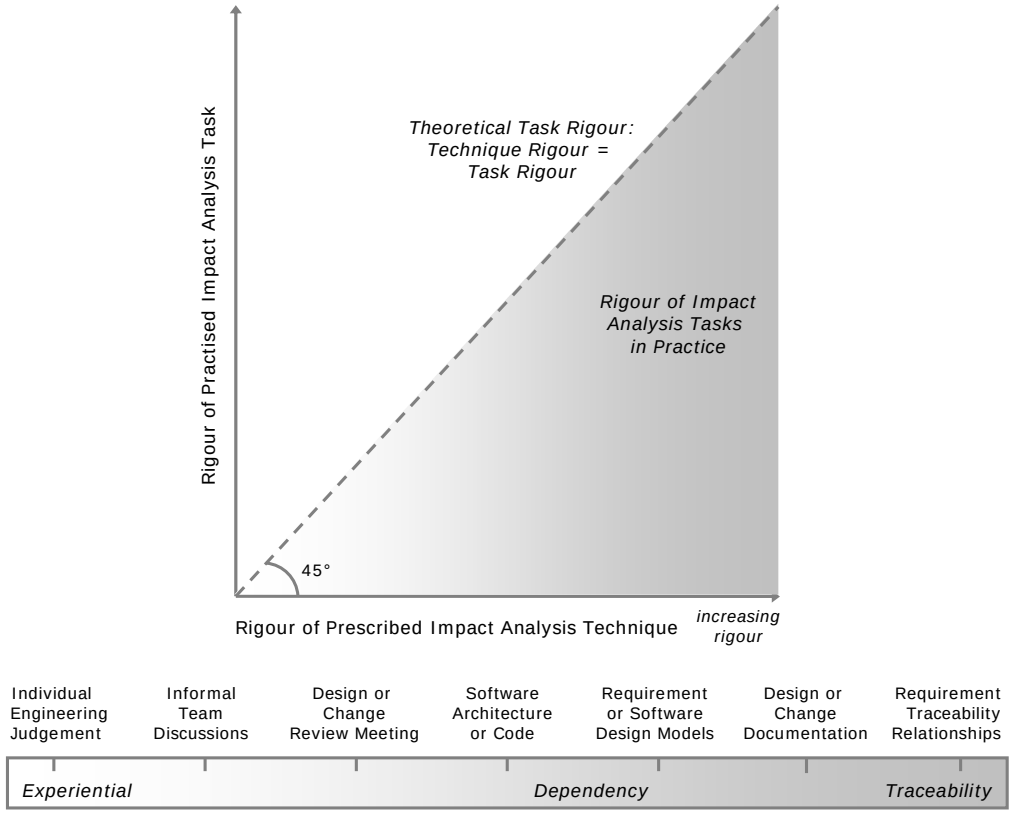

Figure 2. Theoretical, relative rigour of prescribed IA techniques

Figure 2 also illustrates the relationship between IA tasks and techniques in terms of rigour. The relative ratings of instances of IA techniques do not imply that these techniques are more or less rigorous in practice. The actual implementation of the IA technique affects the rigour of the IA task. For example, engineering judgement may provide a better search for emergent changes than unsearchable system and software design models. Other influences also shape IA tasks in practice, as discussed in the following section. Nevertheless, this characterisation of rigour sets a baseline and perspective for discussing the influences on IA tasks and means to improve the implementation of IA within the design process.

4.2 Influences on IA in practice

The disparity between IA tasks and techniques can depend upon many factors, as elicited during the empirical studies. The responses from designers have been categorised into *technique* and *task* influences. Technique influences include the difficulties encountered during IA due to the implementation process or method of the technique, while task influences inhibit the quality of the IA results due to the context of the application of the technique. In other words, technique influences can systematically and repeatedly affect the results produced by IA techniques as implemented within a change process, and task influences affect specific change cases independent of the change process implemented. Technique influences can decrease the rigour of a particular instantiation of an IA technique or may even preclude a designer from applying the IA technique because no search on possible change propagation paths can be performed. In turn, task influences may only limit the search space of the IA technique, affecting the quality of the IA results. Figure 3 summarises these influences as elicited in the empirical studies.

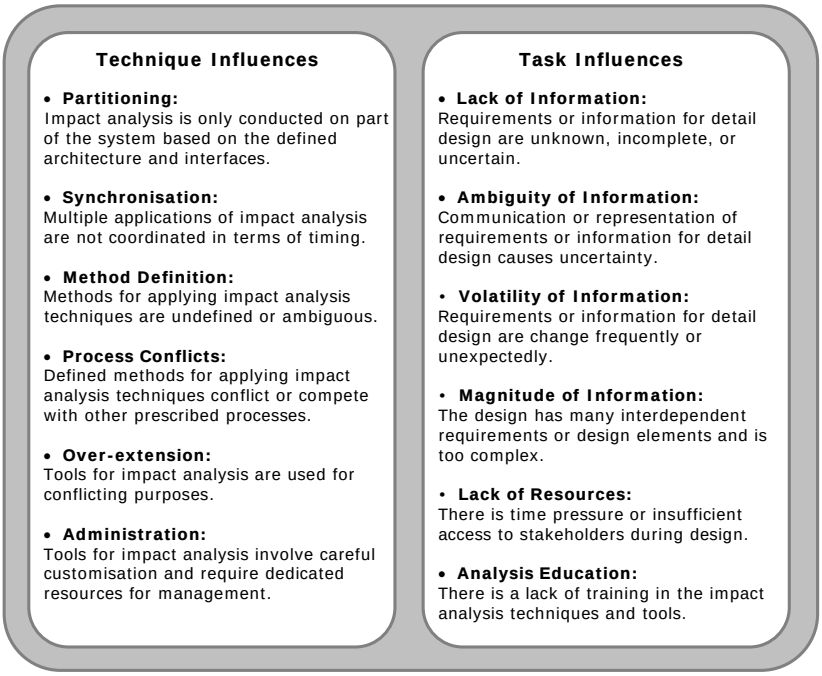

Figure 3. Influences on IA tasks and techniques

4.2.1 Technique influences

A degradation of IA result quality can systematically occur due to a variety of factors affecting IA techniques.

Partitioning can affect the quality of IA results when different system or software designers cannot always analyse the implications of changes on the other area of the design. For instance, systems and software engineers may not have the tools or information sharing capability within the change process to rigorously perform IA on both domains in a timely manner. A single designer may be required to make decisions on design changes affecting both areas using partial IA due to time or design constraints. Thus, the division of the design into these sections can limit the IA search space.

S*ynchronisation* of modifications can be difficult due to the distribution of the design work between systems and software engineers, similar to the partitioning influence. Multiple changes from different stakeholders can affect a single design element or functionality. In some cases, these modifications can compound or contradict each other, leading to unexpected rework to the element or multiple elements producing a single functionality, without a process for coordinating upcoming modifications and IA tasks.

Method definitions may not exist for IA implementations, in some cases, when the process of implementing an IA technique is not well defined. For example, traceability tools may be deployed without prescribing how or when to perform IA, leading to disuse of this IA technique. Fundamentally, the IA technique lacks integration into the change process.

Process conflicts for an IA technique, in contrast, may be systematically caused by multiple defined and prescribed procedures. For example, configuration management procedures may inhibit traceability IA within a change process. A prescribed configuration management process may require that only formally accepted changes are entered into a traceability database. Hence, the traceability relationships captured are not always up-to-date since change authorization can require a significant amount of time and the design may have progressed in the meantime. The actual traceability relationships may differ from the captured relationships in the database, potentially causing incomplete IA results from the database. Similarly, a lag in updating models or code used in dependency IA can affect the quality of IA results produced.

Over-extension of tools also used for IA may also contribute to a decrease in IA result quality. For example, design traceability may only be captured within one database tool. Thus, this repository of information may be the only means to get a high-level overview of the impact of a change. However, if this tool is also used for configuration management, users may use the tool for some IA and then also be required to manually search through more up-to-date requirement documentation not captured yet inserted into the tool. Thus, human error may decrease the theoretically quality of the IA results.

Administration of some tools to implement an IA technique may be a substantial effort. Tools may require customisation or dedicated resources for constant maintenance in order for use. In these instances, the overhead cost of the tools may outweigh the IA results produced, leading to disuse of the tool and IA technique.
These technique influences can preclude the process of implementing IA techniques. Within a change process, these factors can consistently reduce the quality of IA results produced or lead to disuse of the IA technique.

4.2.2 Task influences

In contrast, task influences can affect the quality of IA results despite of the IA technique or change process performed. Many of the responses from the empirical studies focused on the information available in order to perform IA as problematic.
Lack of information regarding new design requirements or changing specifications to use as a basis for the traceability, dependency, or experiential IA was cited by designers. In this case, many different change propagation paths are possible since the design modification is essentially undefined. This uncertainty of the scope of the change can lead to poor quality IA results.
Ambiguity of information or descriptions of design changes can cause similar difficulties in performing IA since the areas affected by the modification are difficult to pinpoint based on the change details given. If IA is implemented with ambiguous information, the IA results may be incomplete or unclear.
Volatility of information due to frequent or unanticipated changes to requirements or detail design information was also found in the empirical studies to be perceived by designers to affect the quality of IA results. As many design changes occur either simultaneously or in rapid succession, the state of the design is rapidly evolving. Designers highlighted the difficulty in considering the compound impact of these changes with their anticipation of further, unexpected changes, ultimately resulting in a low quality of IA.
Magnitude of information, in contrast to the lack of information influence cited, was pointed to by designers as a factor decreasing the quality of IA results. In these instances, the complexity of the design often makes rigorously searching all possible paths of change propagation difficult. The knock-on effects of making a change to a highly coupled design can be difficult to identify in this case, and, consequently, the impact analysis results may not be complete.
Lack of resources , in terms of a lack of time to do a design change or people to perform or have input into a design change, also can affect the quality of IA results. The time pressure of completing a design can cause a rush to simply implement changes without performing any IA or only quickly using engineering judgement and intuition. Some designers discussed the time pressure experienced during the design process, and they were forced choose between producing code as opposed to other design artefacts, such as rationale or traceability documentation and models. This focus on code can cause a scarcity of other design artefacts used for IA. Thus, the IA performed may be of low quality due to the incomplete nature of the inputs to the analysis or since these artefacts may not be up-to-date. Similarly, designers indicated that a lack access to stakeholders limits the inputs to IA. Without understanding the perspectives of all stakeholders on a design change, any IA performed may be incomplete.
Analysis education, or a lack of training, was finally suggested by designers as a source for poor IA result quality. Training may not be delivered in a timely manner in these cases, limiting the IA technique implementation.
Despite these factors that can decrease the quality of IA results, leading to unexpected, emergent modifications, task influences do not always preclude the implementation of some type of IA. IA can still be performed even with imprecise or some unavailable information. For instance, experiential IA does not require significant resources or training. The perception that these influences halt the implementation of some IA techniques, as observed to be believed by designers in the empirical studies, is not the case; there may only be a degradation of result quality.

5 SCENARIOS FOR IMPROVING IA TASKS

Given the characterisations on IA rigour and influences presented in section 4, the following discussion analyses several methods to improve IA tasks elicited in both companies participating in the empirical studies. Often, interviewees only suggested one of the following IA improvement

scenarios; only on a couple of occasions did a study participant cite more than one strategy. This research suggests that addressing each technique and task influence within a specific change process can provide a holistic strategy to reduce unexpected, emergent changes.

5.1 Include more IA

When queried on how to improve the quality of IA task results, designers implementing change management from either a systems engineering or agile software development perspective (as observed within the aerospace and telecom companies, respectively) recommended performing "more" IA to improve the quality of the IA results. These suggestions frequently occurred in the context of performing some form of traceability or dependency IA as opposed only to using an individual's engineering judgement. Some other strategies suggested improving the design review process, entailing more frequent reviews or having more stakeholders or experts review the design. Other research has identified similar suggestions [13].

In terms of the characterisations developed, these descriptions of including further means for IA essentially imply the addition and application of more rigorous IA techniques. Introducing traceability or dependency tools and techniques above and beyond the typically used experiential IA increases the theoretical rigour of the IA tasks performed, as shown in Figure 2. While more rigorous IA techniques can make the search process for potential paths of change propagation more systematic, more rigorous IA techniques do not necessarily inherently reduce the risk of emergent changes due to the influences on implementing these methods in practice. Without addressing some the influences summarised in Figure 3, prescribing the use of more rigorous IA techniques can produce the same quality of IA results as prior to the change in policy.

5.2 Improve quality of inputs to IA tasks

Alternatively, designers interviewed in the empirical studies assessed that the quality of information for the IA task primarily affects the quality of results. Although the design process is inherently iterative, getting the requirement "right-first-time" can reduce the rework necessary. They suggested that improved and timely communication between stakeholders could improve the information available for IA. For example, systems and software engineers may work asynchronously; System designers may write and analyse a set of software requirements and only hand over these specifications to the software designers once they have reached a fairly steady state. The software design can then begin the detail design from the requirement set. As the software detail design occurs, the system designers begin on the next set of requirements. Changes found during the software detail design are often then communicated to the systems engineers. However, since these changes cannot be implemented instantaneously due to resources and time constraints, the details of the changes can be forgotten or IA performed and documented at the time of the change identification can become out-of-date as the design progresses, leading to a ripple of changes during the design process. Implementing just-in-time IA using current design information can capture knock-on effects from other changes that have occurred later in the design process. Furthermore, the communication of these results to all stakeholders can eliminate knock-on effects during the other IA tasks performed within other areas of the design.

In some cases, designers also suggested that allowing for more time and resources during the IA task can improve the IA results. Some designs may be rushed for a software release in order to meet planning milestones with a known degradation of design quality (e.g. software errors). These scheduling constraints can lead to additional, unexpected design changes and iterations, and changing the planning of projects can address such issues.

These suggestions primarily concentrate on the task influences as classified in the categorisation presented in Figure 3. Technique influences can still arise in more strict implementations of systems engineering or agile software development design processes, causing emergent modification to occur.

5.3 Improve IA techniques

Finally, some interviewees suggested improving the specific techniques used for IA. These comments often occurred to improve the designer's favoured IA technique. Ideas to improve the access to traceability and dependency IA the tools were suggested. For instance, some designers suggested having more software licenses available for personal computers. More significant modifications to the implementation of the IA technique were also found. In these cases, designers offered to build larger

and more detailed models of the system for analysis. These proposals occurred for both traceability and dependency IA. More data on the nature of the traceability relationships could be included in the model (e.g. design rationale); or, more detail could be included in the models of the high-level system functionality.

These proposals essentially focus on eliminating some of the task and technique influences. Larger and more detailed models can specifically address the factor of system partitioning. Theoretically, an all-encompassing model could describe an entire product at both a high and low-level of granularity. Improving the ability of a model to capture specific, required information can also take into account the information perceived to be unavailable, ambiguous, or not required and manage the magnitude of information. Furthermore, such improvement can inherently allow for a broader search space for paths of change propagation, increasing the rigour of the IA technique. However, the construction of such models can still have limitations in their implementations within IA processes due to other influences on IA techniques. The integration of developing these models within a design process also must be considered.

6 IMPLICATIONS OF SCENARIOS FOR IMPROVING IA TASKS

The discussion of the scenarios illustrates that there is no panacea to improving the quality of IA results, and, thus, unanticipated change propagation. Based on the characterisations developed from the empirical studies, this research aims to understand how the task and technique influences to implementing IA can be addressed and proposes that understanding how IA tasks are implemented within change processes must first be addressed. As opposed to developing a new technique for IA, by first systematically baselining the current IA tasks applied in practice, directions for improving the process of implementation of these techniques can be identified. This approach may allow for a more holistic strategy to particularly dealing with the technique influences (i.e. process conflicts and synchronisation), thereby, increasing the IA task rigour and reducing emergent changes.

While a broad range of possible influences has been covered in the previous discussion, the empirical studies indicate that all of these influences do not necessarily occur simultaneously for specific change instances within the design process. Depending on the nature of the modification (e.g. a high-level requirement change vs. a detail change in software code), different IA techniques are selected and can equally produce high-quality IA results. Similarly, IA tasks may be provided with enough information, resources, and time. The context of performing IA within an instantiation of a change process with various influences enabled affects the IA results. On-going research is analysing the data collected during the second phase of the empirical studies, including the inputs to IA tasks (quality of information, resources available, and time to perform the IA), the influences in selecting the tasks, and the quality of IA results. This research proposes that investigating patterns of changes and IA tasks in detail can help to identify the detailed trade-offs between the inputs and influences on IA tasks and the quality of the IA results. In turn, the task and technique influences typically associated with different types of changes can be determined. A method for extracting the patterns is being developed to highlight the trade-offs in selecting and implementing IA tasks, allowing for a conscious choice when prescribing or practicing IA within a change process.

7 CONCLUSION

Identifying means of change propagation through rigorous IA can reduce unexpected modifications in the system and software design to occur. The characterisations on IA rigour and influences through the empirical studies provide a perspective to analyse IA within a change process. Analysing the elicited scenarios for improving the quality of IA results indicates the difficulty in developing a strategy for process improvement. On-going research attempts to mitigate these difficulties and take a more holistic approach by first determining the set of task and technique influences on selecting IA techniques for specific change types, thereby developing a focused understanding on the causes for the disparity between IA tasks and techniques.

REFERENCES

[1] Haskins, C., Ed., “INCOSE Systems Engineering Handbook”, INCOSE, 2006.

[2] Maier, M. and Rechtin, E., “The Art of Systems Architecting”, Boca Raton, CRC Press LLC, 2000.

[3] Bohner, S. and Arnold, R., Eds., “Software Change Impact Analysis”, IEEE Computer Society

Press, Los Alamitos, 1996.
[4] Clarkson, P. J., Simons, C. S. and Eckert, C. M., "Predicting Change Propagation in Complex Design", ASME Design Engineering Technical Conferences, Pittsburgh, paper no. DETC2001/DTM-21698, 2001.
[5] Goulding, C., “Grounded Theory: A Practical Guide for Management, Business and Market Researchers”, London, Sage Publications Ltd, 2002.
[6] Dick, J., “Design Traceability”, *IEEE Software*, 2005, 22(6), 14-16.
[7] Briand, L., *et al*., “Automated impact analysis of UML models”, *The Journal of Systems and Software*, 2006, 79, 339-352.
[8] Agile Manifesto, Website, 2001.
[9] Ambler, S., “Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process”, New York, John Wiley & Sons Inc, 2002.
[10] Lindvall, M., Komi-Sirviö, S., Costa, P. and Seaman, C., “A State of the Art Report: Embedded Software Maintenance”, Fraunhofer Center for Experimental Software Engineering Maryland and The University of Maryland, College Park Maryland USA, 2003.
[11] Graaf, B., Lormans, M. and Toetenel, H. (2003). “Embedded Software Engineering: The State of the Practice.” *IEEE Software*, 20(6), 61-69.
[12] Lindvall, M. and Sandahl, K., “How Well do Experienced Software Developers Predict Software Change?”, *Journal of Systems and Software*, Vol. 43, No. 1, 1998, pp. 19-27.
[13] Chao, L., Tumer, I. and Ishii, K., “Design Process Error-Proofing: Engineering Peer Review Lessons from NASA”, *ASME Design Engineering Technical Conference*, Salt Lake City, Utah, 2004.

Contact: M. S. Kilpinen
University of Cambridge
Department of Engineering
Trumpington Street
Cambridge CB2 1PZ
UK
+44 (0) 1223 332828
mk432@cam.ac.uk