INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

ASSESSING IMPACT ANALYSIS PRACTICE TO IMPROVE CHANGE MANAGEMENT CAPABILITY

Malia Kilpinen[1], Claudia Eckert[2] and P John Clarkson[1]
(1) University of Cambridge, UK (2) Open University, UK

ABSTRACT
Impact analysis (IA) methods and tools support designers in determining the consequences of design changes. As such, the risk of unanticipated propagation of changes, in which an initiating modification induces a set of additional, knock-on changes, can be managed through obtaining high-quality IA results that are complete, correct, and clear. Although a variety of IA techniques are proposed in academic publications and even implemented in commercial tools, the use of such IA may not be consistent in practice and can be influenced by the product development process. This paper reports on a method to systematically elicit and assess IA practice in systems and software engineering to identify trends and gaps in change management. Qualitative and quantitative results from the application of this method at an aerospace company are discussed, leading to the proposal of future work to support and extend the method.

Keywords: Change impact analysis, change propagation, rework

1 INTRODUCTION
Understanding the consequences of design modifications is a key element to effective change management. Without grasping the scope of modifications, changes can often unexpectedly propagate and lead to rework, affecting product development budgets and planning time scales. As such, change *impact analysis* (IA) techniques have been developed to support designers in scoping modifications. For example, the Change Prediction Method (CPM) tool calculates and visualizes the consequences of design modifications using a risk-based assessment [1]. Other methods in systems and software engineering, which can be classified into three types, including *traceability*, *dependency*, and *experiential*, similarly aid designers in determining the effects of changes [2].
However, the application of such IA methods and tools is not always promoted or prescribed within companies, and, subsequently, IA techniques may not be systematically used in practice. In turn, the quality of IA results can vary directly, given that IA techniques range in rigor; some IA techniques may not as exhaustively search for high-order knock-on effects as others, and the corresponding IA results may only indicate fractions of the necessary changes stemming from an initiating modification. Alternatively, if IA tools become disused by some design team members during product development, the information they contain can become irrelevant and out-of-date as the design evolves. In this case, the tools unknowingly may produce unreliable IA results when used, indirectly causing variation in the IA quality within a company.
Understanding the available IA methods and tools (Section 2) and how designers use them is integral in assessing a company's capability to manage design changes. Based on observations during an empirical study at an aerospace company (Section 3), a method was devised to systematically elicit and analyze IA practice within their software design process (Section 4 and 5). This method entails identifying the IA techniques used for different types of changes and determining the influences affecting the IA results obtained. Investigating the trends in this data collected depicts the strengths of the company's IA capability and areas for improvement. Quantitative and qualitative results from applying this method within the aerospace company are discussed (Section 6). This paper concludes by proposing how this method can be supported through change databases and suggests further work on how to assess a company's change management capability (Section 7).

2 CHANGE IMPACT ANALYSIS

In previous research into change IA for software-intensive systems, we identified three primary categories of IA techniques, including *traceability*, *dependency*, and *experiential* IA [2]. Figure 1 gives examples of each of these types of IA from literature, which are discussed below. This classification extends the previous categorization of IA by Bohner and Arnold [3], which only recognizes traceability and dependency IA.

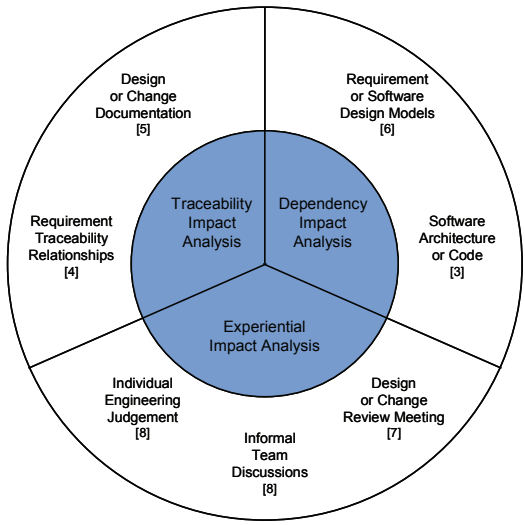

Figure 1. Types of IA techniques

Traceability IA uses the mapping of product requirements to their respective detail specifications and designs within subsystems (e.g. system requirements are linked to more detailed software specifications, which also trace to software design artifacts, such as models, documentation, or code files). Given a change to a system requirement or design artifact, the captured traceability relationships extending from the affected element can be identified. These traces, which also facilitate completeness in product design documentation, indicate the possible system requirements, specifications, or design areas to which a change may propagate. Several academic [4] and commercial computer tools (e.g. Borland Caliber-RM™ and Telelogic DOORS®) support the automation of this style of IA by structuring a database of traceability relationships among requirements and design artifacts. Other forms of design or change documentation not organized in such a database can also be used for IA by manually searching references between and within documents [5]. However, this search process can be difficult to do exhaustively for complex designs.

Dependency IA is another means to investigate the consequences of changes. In software, capturing linkages between variables, logic, modules, etc. within software models and code allows for a detailed analysis of the knock-on impact of a modification. The dependencies identified in software design models [6] or actual code [3] can be used to uncover additional design areas affected by an initiating change. This IA may be automated or performed manually, but tends to focus on specific, low-level changes rather than high-level modifications, which traceability IA could assess. Similarly, the CPM proposed by Clarkson et al. [1] has been applied to mechanical systems to perform dependency IA by analyzing the linkages between system components.

Experiential IA is an additional style of determining the impact of a design modification. Engineers may perform experiential IA by discussing a proposed modification within the design team and using their collective [7] or individual [8] understanding and experience with the design to estimate the consequences of the change. As such, experiential IA can identify tacit design 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, it does not exclude the implementation of traceability and dependency IA, and vice versa.

3 EMPIRICAL STUDIES

The entire range of IA techniques depicted in Figure 1 was observed to be available for software development at an aerospace company during an empirical study. In this firm, embedded software implements the algorithms and logic of several electronic controllers within predominantly mechanical products. These controllers interact with mechanical actuators and electronic sensors to manipulate the product and ensure safe operation. Given that the control system platform is adapted to extend the aerospace company's product line, as opposed to being designed from scratch, design changes drive their software development process. As such, supporting a range of complementary IA techniques can enable performing thorough change assessments to obtain high-quality IA results, which reflect necessary knock-on modifications and rework, and, consequently, augment the company's ability to effectively manage changes throughout the product development process. During the early phases of development, changes tend to be assessed through experiential IA when most information is not yet captured in formal documentation and databases for traceability and dependency IA. As projects progress, traceability and dependency IA can more readily be used.

The first, exploratory phase of the empirical study at the aerospace company aimed to understand IA as prescribed and practiced within software development. In 26 semi-structured interviews with designers, managers, and design process engineers, lasting between 1 and 2 hours each, the circumstances surrounding the application of IA techniques were discussed. However, an early observation illuminated the fact that interviewees could not systematically and objectively discuss the use of IA. They would often fixate on their preferred techniques rather than covering the range of IA techniques used. Furthermore, typical IA practice could not be extrapolated or data mined from other sources, given that the implementation of IA tasks within the software design process was not specifically prescribed and no record was kept regarding the use of these techniques in practice (e.g. in a database that logs proposed engineering changes). Similar difficulties were also encountered during an empirical study at a telecommunications company, consisting of 9 additional semi-structured interviews, lasting approximately 1 hour each, and a 3-day design workshop. Consequently, a method to systematically discuss with system and software designers the IA techniques applied and the influences affecting the IA results obtained was developed (Section 4) and implemented in the second phase of the empirical study within a project group at the aerospace company (Section 5).

4 A METHOD TO ASSESS IMPACT ANALYSIS PRACTICE

During the first, exploratory phase of the aerospace study, designers were observed to be able to readily discuss the IA techniques used for specific changes. As opposed to describing IA practice in general, they could recall and report their actions in these particular situations. As such, the method developed systematically elicits the IA techniques applied to specific instances of system and software design changes. Accordingly, a range of different types of changes must be elicited to enable a holistic view of IA practice. For example, if only high-level requirement changes and the associated IA implemented are discussed with designers, the data collected will skew the perception of IA practice obtained. The method developed characterizes different types of changes according to *attributes*, including *source*, *direction*, *level*, *formality*, and *timing* (discussed in Section 4.1). Different changes have different ratings for each of these attributes, and discussing changes with the range of attribute rating permutations allows for systematically covering IA practice [9].

Besides determining the IA techniques performed and given that the method developed aims to analyze the effectiveness of IA practice, the quality of the IA results obtained also should be estimated, and additionally eliciting the *influences* affecting these results augments this assessment [2]. Thus, the method consists of 3 primary data-collection steps conducted through interviews with system and software designers to elicit a range of specific changes each with different attribute ratings:

1. Rate a change's attributes (Section 4.1),
2. Elicit the associated IA techniques used (Section 4.2), and
3. Estimate the IA result quality and determine the influences affecting these results (Section 4.3).

The method then analyzes IA practice by comparing the 3 categories of data collected in each of these 3 steps against each other (Section 4.4). These steps are detailed in the following sections as employed for the aerospace company's development process. Nevertheless, this method is envisioned to be applicable to other companies in that other firms even in different industries similarly use a

systems engineering approach to software development and make a variety of IA techniques available. However, customization of the attributes (Section 4.1) may be required for such application in order to appropriately describe the specific context of system and software design changes.

4.1 Step 1: Rate a change's attributes

A change's *attributes* (detailed in [9]), including *source*, *direction*, *level*, *formality*, and *timing*, are derived from asking the following corresponding questions:

- **Source** – Who requested the change?
- **Direction** – Who implemented the change?
- **Level** – What was the change requested? / Where (on what level) was the change implemented?
- **Formality** – How was the change requested?
- **Timing** – When was the change occurring in the context of other changes?

Responses to these questions by designers can characterize noteworthy aspects of a design change, which, in turn, can be correlated with the IA techniques implemented. The *source*, *direction*, and *level attributes* aim to capture if the change was requested across an interface and on what level of detail the change was actually implemented. Certain IA techniques may be more suited for changes contained within software code, while others may be more appropriate to estimate the consequences of high-level requirement modifications occurring across both the system and software designs. As such, these questions account for the difference in IA techniques applied for software code versus system requirement modifications, for instance. The *formality attribute* describes the process by which a change was requested (e.g. through a formal, documented change request or only in passing discussion between designers) and suggests any barriers preventing the use of certain IA techniques. For example, as observed in the empirical studies, a lack of documentation of a change often can limit the usability of traceability and dependency IA since insufficient information about the modification is provided for input into these analysis methods. Finally, the *timing attribute* question determines the existence of interdependent changes and, correspondingly, estimates the comprehensiveness or rigor of IA required for high-quality results.

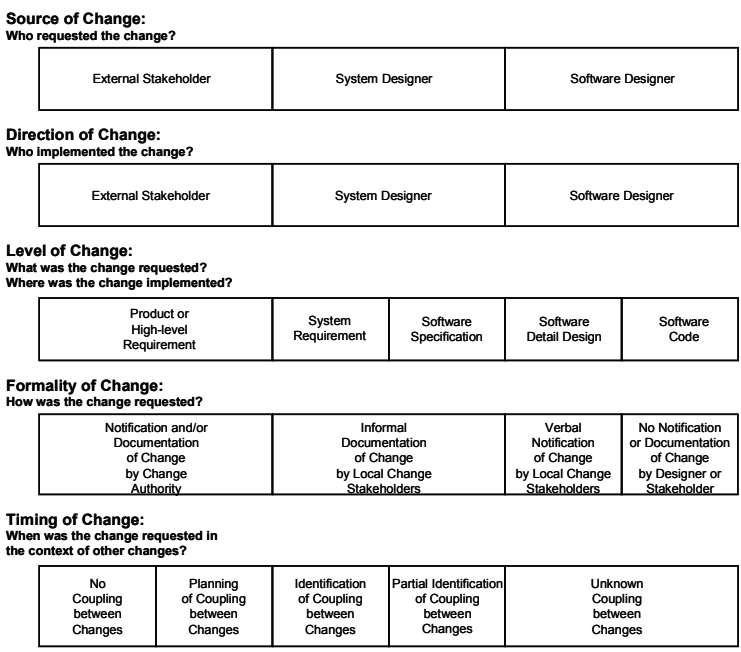

Figure 2. Categories for source, direction, level, formality, and timing attributes

Based on the first, exploratory empirical study phase, *categories* to qualify the designers' responses for each of the above attribute questions were constructed for the aerospace company. For example, the *categories* of the source attribute consists of *external stakeholder*, *system designer*, and *software designer* (in Figure 2) and suggests the expected range of responses. (In this case, external stakeholders include the company's mechanical and hardware designers as well as customers of the product, all of whom are external to the software design team.) The collection of many examples of changes in the second empirical study phase supported the refinement of the categories and their descriptions. Figure 2 depicts the final categories for the source, direction, level, formality, and timing attributes. Notably, the source and direction attributes use the same categories since the stakeholders shown capture all possible change requesters and implementers. These categories are envisioned to require some customization for other companies using similar systems engineering development processes, but with alternative team structures and interfaces than the aerospace company.

To complete step 1 in the method and qualify a change's attributes, the interview with a designer should cover all of the questions listed above. The responses gathered should be correlated with the categories depicted in Figure 2. Figure 2 essentially provides a map to locate specific responses for each attribute across the range of potential responses. Designers' responses can then be quantified based on their categorization and given ratings of high-medium-low (Figure 4). Thereby, the spectrum of different types of changes elicited, as exhibited by different attribute rating permutations, can be assessed (discussed in Section 5).

4.2 Step 2: Elicit the associated IA techniques used

For each specific change discussed, designers should be prompted to walk through each step of their involvement – from change proposal to implementation to rework, if required. A list of IA techniques available in the company should be available to the designer during this recollection for him or her to highlight those techniques he or she used. The IA techniques applied can then be noted.

4.3 Step 3: Estimate the IA result quality and determine the influences affecting these results

In the first, exploratory phase of the empirical study, influences affecting IA results were identified by coding and analyzing interview transcripts (detailed in [2]). Figure 3 illustrates the influences classified through this process. *Technique influences* systematically affect the process of applying IA, while *task influences* are primarily caused by situation-specific limitations on information, resource, or time availability.

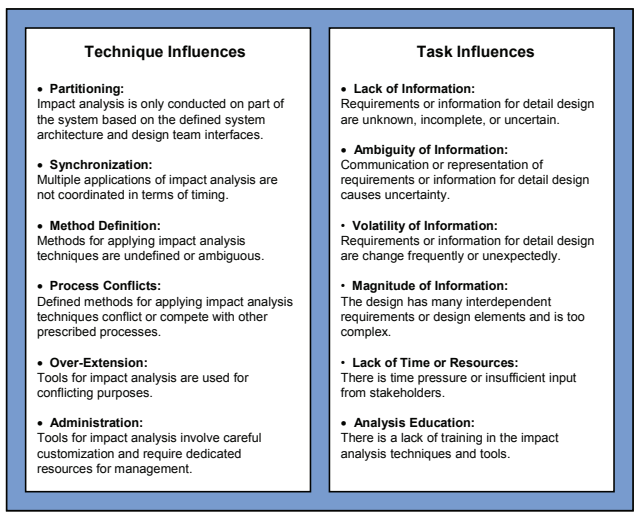

Figure 3. IA influences

Asking interviewees to rate the information, resource, and time availability on a high-medium-low scale (e.g. no missing information, some missing information, or little information available, respectively) and describe the rationale for this rating induces discussion about any IA task influences present for a change. The designers can then be asked to estimate the quality of IA results obtained for the change in light of these influences. This quality estimation can be captured by scoring the result quality as high, medium, or low (i.e. no emergent changes expected, some emergent changes expected, or a significant number of emergent changes expected, respectively) and should reflect the overall understanding of change impact if more than one IA technique is used.

Invariably, IA technique influences can also be derived through the discussion of their ratings of information, resource, and time availability, given that even high information, resource, and time availability can still produce low-quality IA results because of the change process implemented. For example, there may be enough information, resources, and time for a requested change; however, the traces or dependency relationships in communal databases for IA may be out-of-date, systematically causing incomplete, low-quality IA results. In this case, the IA applied could be affected by the *process conflicts* influence (Figure 3) in that the database could also be subjected to configuration management procedures, which only allow formally accepted changes to be entered into the database. Hence, the captured relationships are not always up-to-date since change authorization can require a significant amount of time as design work progresses. The configuration management procedures conflict with the process of implementing IA necessary to produce high-quality results. As such, prompting designers to provide additional, hypothetical ratings for IA result quality and their rationale for this prediction given hypothetical increases in information, resources, and time availability ratings can support this elicitation of IA technique influences.

Therefore, to complete step 3 in the method, a change should be rated on a high-medium-low scale for information, resource, and time availability as well IA result quality and the rationale for these rating discussed. Then, the information, resources, and time parameters should each individually be changed from their original scores to high ratings (if they were not already scored as high), inducing additional hypothetical ratings of the IA result quality by the designers. Then, all of the parameters should be set to high, and the interviewees should provide a rating for the expected IA result quality and their rationale in this optimal scenario. This rating exercise provides a basis for discussion from which the IA influences present can be noted. Notably, the subjectivity of interviewees affects the ratings obtained. However, given that the discussion of the reasoning behind these ratings is used to derive the IA influences present and their frequency of occurrence, the subjectivity of the ratings does not directly affect these results obtained, but facilitates the discussion of IA influences.

4.4 Step 4: Compare data collected in previous 3 steps

The 3 categories of data collected (the change attribute ratings, the IA techniques applied, and the IA quality and information-resource-time availability ratings) in the previous 3 steps can be compared against each other to identify trends and make observations on IA practice. Specifically, a category of data can be compared against another by mapping the similarities and differences in how elicited changes split across the specific data fields collected in each category. In this manner, three comparisons can be made:

- The IA techniques applied vs. the change attribute ratings
- The IA techniques applied vs. the IA quality and information-resource-time availability ratings
- The change attribute ratings vs. the IA quality and information-resource-time availability ratings

In addition, the IA techniques applied can be compared against the range of IA techniques available to make a final observation on the frequency of use of these techniques in practice.

5 METHOD APPLICATION AT THE AEROSPACE COMPANY

Elicitation of IA practice in the aerospace company occurred through discussing specific instances of changes with 6 systems engineers and 7 software designers within a software development project group. At the time of these interviews, the project only consisted of 17 system designers and 20 software engineers. Thus, about 35% of both the system and software design teams participated in this IA practice study. These designers, as opposed to external stakeholders, who primarily requested changes, were targeted since they actually employed the IA techniques for software modifications.

The interviewees discussed changes they personally worked on implementing, and, thus, they can be considered experts on the IA details of these design modifications.
During the interviews, while all of the attribute questions within the method (i.e. in step 1) were covered for each change discussed, the interviewer classified the responses regarding the type of change alone according to the high-medium-low attribute ratings shown in Figure 4. This reduced the time required for the interview to explain each of the categories and then perform the ratings. The division in the source, direction, and level attributes into high, medium, and low ratings is based on the roles of the system and software designers in the aerospace company and their responsibility for work products. The timing and formality partitioning is created based on the extremes of these attributes as observed in the aerospace company.

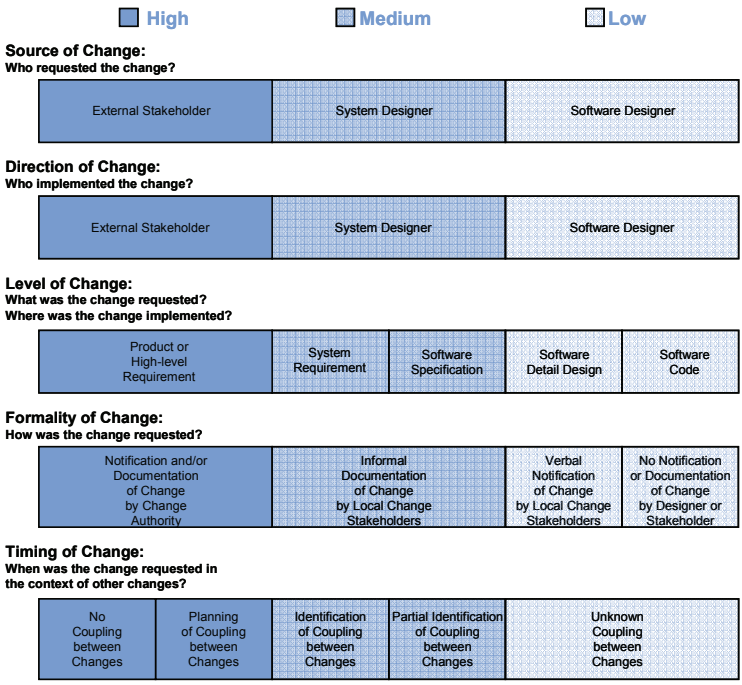

Figure 4. Low-medium-high ratings for change attributes (from Figure 2)

After performing several interviews, the types of changes collected, identified by the variety in attribute permutations, were analyzed in terms of the range of attribute combinations possible, suggesting the breadth of the IA covered in the interviews. Given that certain attribute combinations did not occur, they were inquired about in subsequent interviews. Specific questioning revealed many of the interviewees considered some permutations non-existent given the structure of the aerospace company (e.g. software designers typically do not directly deal with high-level requirement changes), eliminating the total number of possible attribute combinations. Extreme attribute ratings for both formality and timing also did not occur during the discussions and were specifically prompted for in subsequent interviews. However, these design changes with such extreme attributes could not be found. Not many changes occurred without any or with perfect formality or synchronization. As such, this elicitation method presented does not necessarily ensure capturing these extreme cases, but systematically encompasses the range of IA techniques typically applied for changes.
In total, 42 change cases were volunteered by interviewees as examples (50% from systems engineers and 50% from software designers) and discussed in terms of change attributes and IA techniques performed (i.e. step 1 and 2 in the method). Subsequently, only 23 of these changes were rated for IA quality as well as information, resource, and time availability and the rationale for these ratings discussed (i.e. step 3 in the method). Not all of the initial 42 changes were covered because this number of ratings would take a considerable amount of time for each interview. Changes were

selected that included the broadest range of IA techniques reportedly implemented by the interviewee. Occasionally, interviewees suggested replacement changes to cover a broader range of IA techniques than initially discussed for step 3 in the method. In this case, the information, resources, and time parameters were rated, and then the change type attributes and the IA performed of these substitute changes were elicited. These additional changes were not included in the original set of 42 changes since they were not always discussed as thoroughly as this initial set.

6 IMPACT ANALYSIS PRACTICE AT THE AEROSPACE COMPANY

System and software designers at the aerospace company have 11 distinct IA techniques available for their use. These techniques fit into the example categories illustrated in Figure 1. Specifically, traceability IA can be performed through:

- An automated requirement traceability tool,
- Manually following traces in requirement and design documentation, or
- Manually using relationships captured in a data dictionary.

A *data dictionary* is a specialized traceability database that captures the relationships between variables and their use across a design in documentation, models, and code [10]. Given a change to the name, meaning, or use of a variable, the modification can be propagated throughout the design. Dependency IA techniques available to system and software designers include using:

- An integrated software design model,
- Models of individual software design areas,
- Software UML model, or
- Software code.

Finally, experiential IA can occur through:

- Formal design review meetings,
- Integrated product team meetings,
- Informal discussions, or
- Individual engineering judgment.

By comparing the categories of data elicited from the interviewees, observations on how these available IA techniques are used in practice can be identified (i.e. step 4 in the method).

6.1 Observation 1: Several IA techniques are not implemented in practice

By contrasting the IA techniques available and those reported applied to the 42 changes discussed, several IA techniques are observed not to be used. Specifically, the requirement traceability tool, data dictionary, and the integrated software design model were not used for IA (Figure 6). Discussions with designers during the first empirical study phase also suggested the abandonment of these techniques. In the case of the traceability tool and data dictionary, interviewees indicated they did not know that these IA techniques existed, and they also suggested the information in the integrated software design model was too out-of-date to provide useful IA results. As such, this data insinuates that traceability IA nearly always occurs through manually following traces in documentation rather than through the automated traceability software tool. As previously indicated, this style of IA has limitations for complex products since exhaustive searches for knock-on effects can be difficult to perform. In turn, the data collected also reveals that designers most frequently apply dependency IA using the models of individual software design areas and experiential IA through informal discussions (Figure 6), and, overall, experiential IA techniques are applied most frequently (Figure 5).

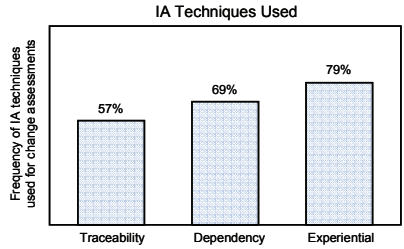

Figure 5. Frequency of IA techniques used

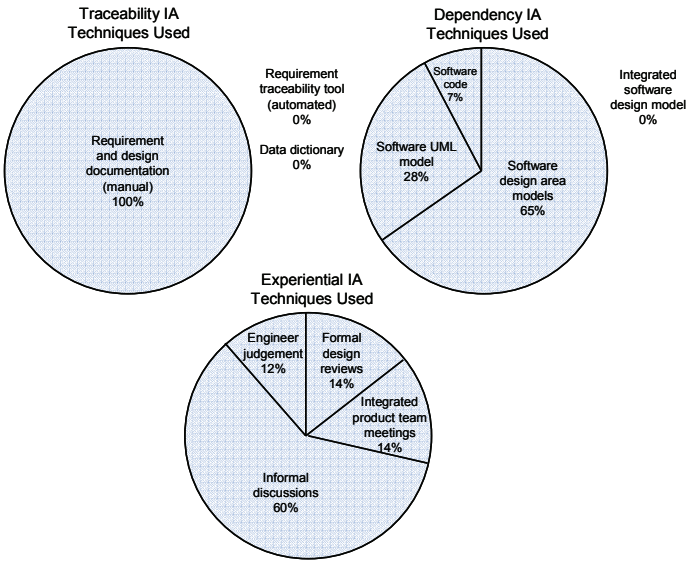

Figure 6. Range of IA techniques used

Based on these trends, improving the range of traceability IA techniques used can improve the company's change management capability, given that without a rigorous understanding of relationships across design areas and their work products (i.e. system requirements and software design artefacts, such as models, documentation, and code) unexpected change propagation can occur. Furthermore, given the strong preference of designers to use models of individual software design areas for dependency IA, advocating the use of the integrated software design model and maintenance of the information it contains that captures dependencies across all design areas can support this aim.

6.2 Observation 2: System and software designers rely on different IA techniques

By cross-referencing the IA techniques applied for the 42 changes against the source, direction, and level attributes of these modifications, patterns of the IA typically applied by systems and software engineers can be identified, as shown in Figure 7. Systems engineers tend to receive change requests from external stakeholders (e.g. mechanical or hardware designers and customers) and modify the high-level software requirements as necessary. They then pass these modifications to software engineers to implement in the detail design. Software designers do not interact directly with external stakeholders due to the partitioning of work between system and software designers, and, given the flexibility of software, the software design generally does not initiate changes in mechanical or hardware designs. Systems engineers only occasionally ask other system designers to implement changes as the high-level software requirements are partitioned into decoupled design areas.

However, the software architecture does not necessarily follow this partitioning, and design areas can become coupled in the implementation of software code, affording means for change propagation.

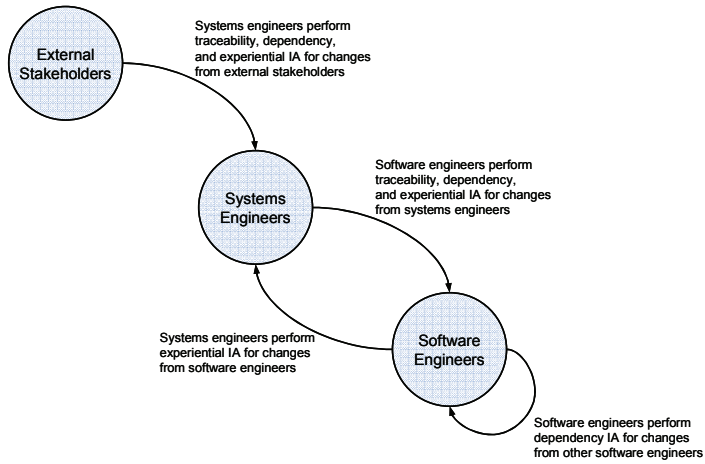

Figure 7. IA techniques used by system and software designers

While traceability, dependency, and experiential IA techniques are all applied by system and software designers (Figure 7), these designers also tend to use different traceability and dependency IA techniques. In particular, systems engineers focus on applying traceability IA with documentation for high-level requirements and do not perform detailed dependency IA with software UML models or code. In contrast, software designers execute traceability IA using the whole range of documentation available, from requirements to detail design specifications, and dependency IA with all of the available techniques (except the integrated model). However, they also tend to concentrate on the detail design and most frequently perform dependency IA. As such, the high-level analysis performed by systems engineers combined with the detail design assessment by software engineers can be very rigorous in that the IA performed can thoroughly identify means for change propagation. Nevertheless, the sharing of IA results between these engineers must be implemented in a timely manner in order for this rigor to be achieved in that knock-on effects and upcoming changes must be accounted for in subsequent IA tasks. Thus, the aerospace company's capability to manage the risk of unanticipated knock-on effects is highly affected by the organizational partitioning of the system and software design roles.

6.3 Observation 3: IA technique and task influences may equally affect IA results

By comparing the IA techniques applied and the IA influences cited by designers, IA technique and task influences (Figure 3) both affect the quality of IA results, despite applying a range of IA techniques. Out of the 23 changes rated in step 3 of the method, designers designated that IA technique influences primarily affected the IA results of 10 changes. In most of these instances, high ratings for information, resources, and time produced mediocre IA result quality and were frequently affected by partitioning and synchronization influences, as shown in Figure 8 and also suggested in the previous section. The remaining 13 changes discussed were predominantly affected by IA task influences, including insufficient information, resources, or time available to perform IA. Designers most commonly cited task influences dealing with information as limiting factors (Figure 8). Given the concurrent nature of the design process at the aerospace company, these ratings fit this expectation of information availability. Even though a relatively small number of changes were rated, the outcome of this rating exercise suggests that both IA technique and task influences can equally play crucial roles in IA result quality and affect the company's ability to manage the risk of unexpected change propagation. The process of implementing IA techniques as well as the product information input into IA should be considered to improve IA practice.

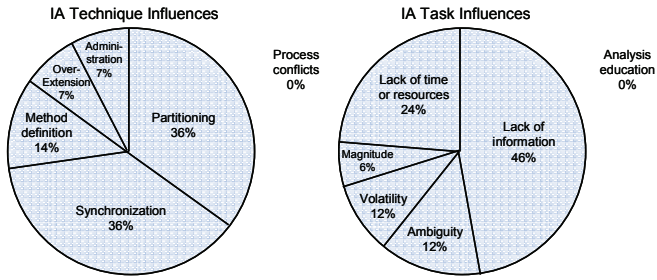

Figure 8. Range of IA influences elicited

6.4 Observation 4: Less than ideal IA results tend to occur in changes requested across interfaces

By comparing the attributes of the elicited 23 changes and their corresponding IA result quality reported and IA influences, changes occurring across interfaces (i.e. the source and level attribute ratings differ) had less than ideal IA results; specifically, the IA result quality associated with these modifications was never rated as high due to the presence of IA influences. Changes requested by external stakeholders tended to cause system designers difficulties in performing IA, while modifications between system and software designers tended to challenge these engineers. These changes were always linked to the partitioning and synchronization technique influences and task influences dealing with information, except for one modification discussed. Consequently, improving practices of obtaining, sharing, and updating information about changes between systems engineers and software designers can address these predominant influences, support high-quality IA results, and improve the company's change management capability. Although the subjective IA quality reported by designers was used to make this observation, analysis of a change database for another recent project at the aerospace company indicates that unexpected, knock-on modifications often occur, suggesting that IA quality is not high in practice. The data collected does not suggest a correlation between the formality and timing attributes and the quality of the IA results or IA influences present.

7 CONCLUSION

The 4 observations made (Section 6) through implementing the method developed (Section 4 and 5) for a software development project group in the aerospace company all suggest that the partitioning of the roles of systems and software engineers affects the quality of IA results obtained. Given that these designers apply different IA techniques, varying perspectives on the impacts of changes are obtained. Without sharing and communicating these results and the associated modifications to be made, upcoming applications of IA can miss knock-on effects. As such, improving the sharing of information about changes by system and software designers allows for improving their change management capability.

Currently, personal communication is the predominant means to supply this overview in the aerospace company; periodic design and change review meetings provide a formal process to discuss modifications, but informal discussions tend to provide the source for up-to-date information about changes and their interdependencies. However, available IA techniques within this company can support this aim. In particular, using the automated traceability tool or integrated software model can allow system and software designers to broadly capture design information and perform IA across design areas, potentially increasing the quality of IA results. As such, a strategy to improve IA could be to improve the use of these techniques. New IA techniques also could be developed to similarly fit this need or even extend or integrate the IA features of these tools. Alternatively, more systematic and frequent processes for communication about and analysis of changes between designers could be implemented to improve IA quality and address the influence of partitioning. While information availability, particularly from external stakeholders, may be difficult to improve in a highly iterative development process for a complex product, improving the information sharing within the software development team in the aerospace company can improve the handling of changes and reduce internally generated rework.

Although the method developed and implemented has provided insight to the IA performed within the aerospace company's software design process, the effort to characterize IA practice could be streamlined. In particular, change databases, which often meticulously catalogue engineering change requests and their impact (i.e. in terms of the product areas affected) and status (e.g. open, closed, or rejected) throughout change processes, could be used to also capture IA techniques used and their results. Additionally capturing such detailed information about the change processes implemented (particularly in terms of IA) could allow for the verification of results from specific IA techniques during product development and also could be used retrospectively to determine the quality of IA results based on the IA techniques used. Improvements to the IA methods and tools and the use of these techniques by designers could then be planned through examining this information on change processes in practice. Assessing data regarding change processes can be as useful as analyzing information on product changes to improve change management capability.

The method presented to assess IA could also comprise a portion of a larger, more comprehensive analysis of a company's change management capability. Besides focusing on IA, other factors affecting change processes can be analyzed. For example, the interfaces across the variety of disciplinary design teams within a company could be investigated in terms of the flow of design modifications. Similarly, the company's product platform also could be examined for changeability in such a comprehensive assessment. Depending on the scale of such assessments and collection of other data on change management practice, the method could be tailored and integrated to meld with these other means of analysis. As such, further work is necessary to identify the aspects of change management practice to investigate and then develop methods to perform such assessments.

REFERENCES

[1] Clarkson, P. J., C. S. Simons, et al. (2004). "Predicting Change Propagation in Complex Design." ASME Journal of Mechanical Design 126(5): 765-797.

[2] Kilpinen, M. S., C. M. Eckert, et al. (2007). Change Propagation at the Interface of System and Embedded Software Design: Characterising Impact Analysis Tasks and Techniques. International Conference on Engineering Design. Paris, France.

[3] Bohner, S. A. and R. S. Arnold, Eds. (1996). Software Change Impact Analysis. Los Alamitos, California, USA, IEEE Computer Society Press.

[4] Dick, J. (2005). "Design Traceability." IEEE Software 22(6): 14-16.

[5] Brown, W. J., H. W. McCormick, et al. (1999). AntiPatterns and Patterns in Software Configuration Management. New York, New York, USA, John Wiley & Sons.

[6] Briand, L. C., Y. Labiche, et al. (2005). "Automated Impact Analysis of UML Models." Journal of Systems and Software 79(3): 339-352.

[7] Endres, A. and D. Rombach (2003). A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories. New York, New York, USA, Addison-Wesley.

[8] Ambler, S. (2002). Agile Modeling: Effective Practices for Extreme Unified Process. New York, New York, USA, John Wiley & Sons.

[9] Kilpinen, M. S., C. M. Eckert, et al. (2007). The Emergence of Change at the Interface of System and Embedded Software Design. Conference on Systems Engineering Research. Hoboken, New Jersey, USA.

[10] Bray (2002). An Introduction to Requirements Engineering. Harlow, UK, Addison-Wesley.

Contact: Malia Kilpinen
University of Cambridge
Department of Engineering
Trumpington Street
Cambridge, CB2 1PZ
United Kingdom
Tel: +44 (0) 1223 748569
Email: mk432@cam.ac.uk
URL: http://www-edc.eng.cam.ac.uk