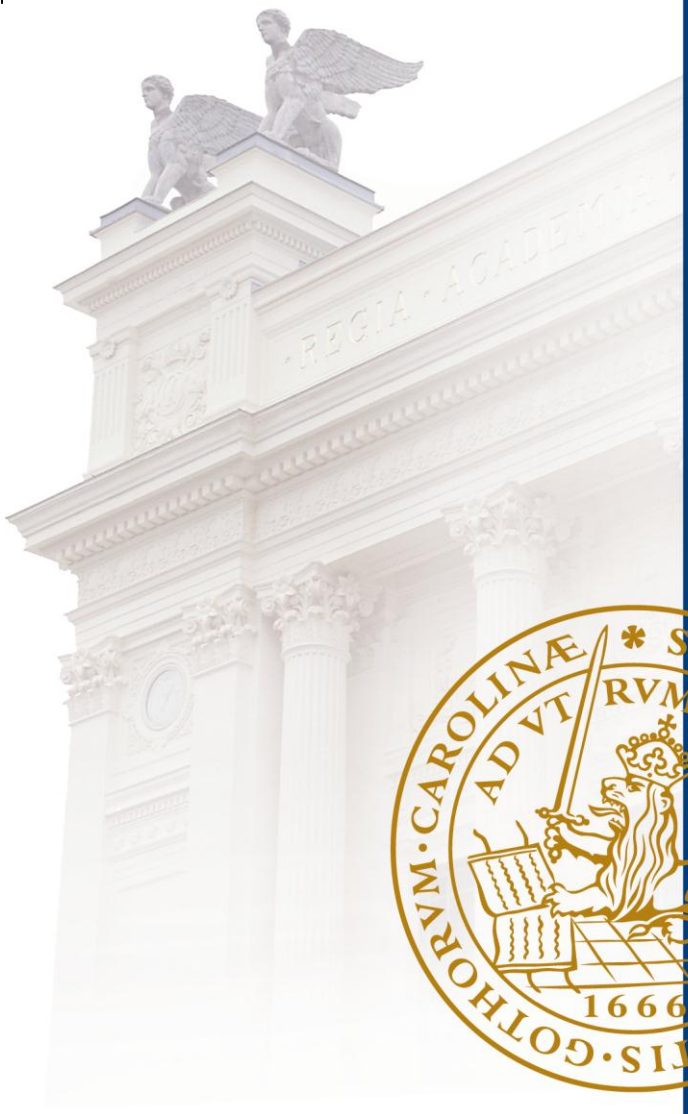
Master's Thesis

People, process and tools: A Study of Impact Analysis in a Change Process

Artour Klevin

Department of Computer Science
Faculty of Engineering LTH
Lund University, 2012



ISSN 1650-2884
LU-CS-EX: 2012-03

People, process and tools:
A Study of Impact Analysis in a Change Process

Artour Klevin

February 19, 2012

Keywords: Manual Impact Analysis, Change Process, Human Aspects of Software Engineering, Software Process Improvement, Human Error, Software Tools, Change Documentation, Safety Critical Software, Change Requests

Abstract
The objectives of this master thesis is to find ways to improve efficiency and quality of the change process, with focus on Impact Analysis. At first a analysis of the working environment is done, to get a better view of the work flow in the change process. This is done to understand the context of the studied site. In this thesis we discuss problems and solutions about how software development can be enhanced with help of: processes, tools, communication, education and dissemination of knowledge in the organization. The problems and solutions are viewed in three domains; people, process and tools. All because it is easy to fall in assumptions that a new tool will solve all the problems. Nevertheless a tool is the first step to create change in a company, with its help trust is gained and problems below surface can be addressed. The main results of this thesis are, a tool that finds simple errors and a process improvement suggestion for the Impact Analysis at the studied site. Also, problems concerning motivation, communication and education are addressed and discussed.

Acknowledgements
We want to thank our supervisor at Lund Institute of Technology, Lars Bendix, for his valuable inputs and feedback on our work. We appreciate the time Lars Bendix have put into guiding us through this master thesis and to inspire us to think outside the box.

Big thanks to everyone at the company for letting us disturb you and ask questions. Special thanks to, Jon Joines, for his guidance and help with practical things at the site. Also, to Ulf Steen, for the long talks about Software Configuration Management and other things.

Thanks to Johan Thiborg-Ericson, for working with me on this master thesis. The difference in our opinions have created many interesting debates and great solutions. The final result would not be as good without Johan as colleague.

List of Terms:
SCM - Software Configuration Management, in this context, has the task of controlling the change and providing a safety net when something goes wrong. It helps to maintain a working product and deliver metrics about its status.
Change Process - The process that should be followed when making a change to the product.
CCB - Change Control Board, a group that can make decisions about change requests.
IA - Impact Analysis, the analysis of technical and economical risk associated with change request. Also, a list of configuration items (requirements, tests, documentation, code, etc) that are affected by the change request. It might also refer to the process of finding those items.
SIL - Safety Integrity Level, in this context, a degree of how rigorous the process for changing a piece of code is.
Merant Tracker - or short Tracker, a tool for change and issue handling in: code, documents, tests, etc.
Tracker Case - The unit in Merant Tracker that contains a Change Request.
Certification organization - a leading international body for the certification of safety and quality for products, services and management systems.
EA - Enterprise Architecture, tool that holds the modules of architecture and connection between requirements, interfaces, design, etc.
CR - Change Request, a formal request about a change to the product e.g. to fix a bug, improve performance or add functionality.
PTT - Product Type Test, test if the whole product can be built.
Impact Analysis form - A list with questions that have to be answered during the Impact Analysis. The version used in this thesis is available in appendix A.

Work breakdown
The thesis work at the company has been done together with Johan Thiborg-Ericson, but the final report is written separately. When writing the code, most work was done by pair programming. The final rapport is carefully checked and reviewed by both, the author and Johan, who helped with ideas and discussions during the whole time this rapport was written.

During the coding Artour had responsibility for working with SQL and creating the GUI, while Johan worked on the parsing of the Impact Analysis and finding the faults in them.

Contents

1. Introduction

The common problem with SCM and processes is that they lessen flexibility and add more time spent on paperwork instead of working on the product. However there are great benefits as well, for example, extra security in the case something goes wrong or ability to collaborate in big teams. There are many things that have to be done: first the process have to be well adapted, second there have to be tools that support the process and finally the people using the process have to understand it. All of those points have to be continuously reviewed and changed to adopt to the changing environment of global businesses. One of the commons faults is that the process is pushed from the top management on to the employees and is not adopted to the way people work. In large organizations with products and processes, that is certified by a third part, changes to the process can be hard to implement. This will make the employees feel that they can't affect how they work. The process has to be developed with the goal to support the users of the process. Therefore there has to be clear communication between the people who forms the process and those who use it. A well adopted process should be a win-win, for both the users and the management. If not, it will be a hindrance and the full benefits of the process might be lost.

Every company developing software have to offer support and maintenance of their products. Maintenance of a complex system is very difficult and costly process, to make it possible to maintain the software techniques like Impact Analysis is used. An Impact Analysis is a list of all deliverables that have to be updated because of a proposed change to the system. It is used to help ensure that the system is performing correctly after the change is implemented and to verify that new fault are not introduced. It is easier to identify where the faults could be introduced, with help of the information in the Impact Analysis. Also to ensure that all vital dependencies are up to date. In the change process the Impact Analysis is a vital part of the process, which helps to keeps the system stable. An analysis of the change process at the studied site showed three phases that where of interest. The first is an evaluation of the Change Request, when the initial Impact Analysis is written by the developer. This is done so that the CCB can decide if the change should be implemented. The second is when the Impact Analysis is revisited by developer after implementation to update the Impact Analysis with better information on the change and reviewed by the CCB to see that all rules have been followed. The third one occurs only for safety products that are developed, especially changes that effect SIL are viewed. The Safety Team reviews the Impact Analyses for a product release and sends them in a report to the certification organization. Which then certify the new product or give feedback on what have to be redone to pass the certification.

The perceived problem in the early stages of the thesis was that there were many simple systematic errors and that there was a need for faster feedback on the Impact Analysis to developers. The CCB also wanted to find simple errors faster and Safety wished for better quality on the information written in the Impact Analysis answers. The focus of the company was a tool as a solution to the problems. Therefore a tool solving the most pressing issues was developed, while at same time informal interviews with developers where carried out.

Initially developers complained about how laborious it was to write the Impact Analysis. But when they have talked about it for longer periods of time other problems got more

focus. These problems were often only vaguely related to writing the Impact Analysis, like the problem with legacy code or that the Impact Analysis questions have grown uncontrollably and that the whole Impact Analysis is now more a burden than something beneficial. Also, there already exists tools for finding the answers to some of the questions in the Impact Analysis, but they are not used much. This has led us to the conclusion that the mentioned problems about the Impact Analysis are just indicators of deeper dissatisfactions. This lead to a wider analysis of the problems, which gave many new problems that had to be addressed. Distributed across all three areas of interest that are covered in this thesis.

In this document it is assumed that the reader has an master in computer science or has work experience from a company developing software for safety products. Only some basic knowledge of configuration management and software process is expected. The theoretical knowledge in configuration management that is necessary for understanding this thesis is provided for employees with no CM-background, in beginning of chapters 2.2 and 2.3. For the students, information about the Impact Analysis questions and the process is describe at length later in the same chapters.

The ambition of this master thesis is to give people who read it, a better understanding of possible ways to efficiently improve software development. Also create some concrete solutions for the site. With help of: better processes, smarter tools, more committed and educated people. This thesis starts with a detailed description of the study site and the process in place. When a better understanding of the companies process is in place, a root cause study is done. As result of that study many problems are presented in a chapter that goes further in to the companies situation, then the initial problem description. The chapter following the analysis is written without the context of the site. Presenting some discussion and solutions about general things in the domain of process, people and tools. In the last chapter solutions and improvements fitting the site are presented.

2. Background

This chapter has been written to supply a better understating of the site and the context of this master thesis. It begins with a subchapter that will describing the studied site for those not familiar with the environment at the site. Next for those interested in the common definition of the Impact Analysis and Change Process the beginning of subchapters 2.2 and 2.3 is recommended. Further in those chapters the Impact Analysis and Change Process at the company is described from the authors point of view. This is done by using information from several documents and interviews with employees at the site. Therefore even people working at the site could find new things to read there. In the end of the subchapters 2.2, 2.3 conclusions of the differences and similarities to the common definitions are made. The subchapter 2.4 describes the philosophy behind this thesis by introducing the triangle concept. In 2.5 the description of how we developed the tool is described. In the last subchapter list of limitations for this thesis is presented.

2.1. Facts about the study site

The code base of the studied product has been developed for many years and is about 20 million lines of code written in at least three different programming languages. It is developed in Sweden, Germany and India. Parts of the code base is reused from old system versions from over 15 years ago. There are about 30 developers working on developing and maintaining this code today. There more than 100 change request every month to the code of which only few changes are deferred. Those change requests create more requests in form of changes to tests, designs and requirements documentation, creating a network of changes to the product as whole and not just the code.

The change requests are stored and tracked in a program called *Merant Tracker* in a so called *Tracker Case*. With this tool CCB can give the developers assignments and follow the status of individual change requests in the different stages of the change process. The information in the cases can be put into the system in two ways. The first way is standardized fields that have to filed in. Some of those fields are mandatory and are filed in by everyone who use Tracker system at the company. These fields are seen by all users of the Tracker system, and thus any change here will have to be approved by many stakeholders. Additional information specific to the workplace can be added in free text notes attached to the Tracker Case, the Impact Analysis is such a note. Other notes are added to the cases during the change process holding information about test results or important decisions.

Some years ago the company started to develop according to a safety process and they are now one of the few companies in the world that has a process and product that lives up to the IEC 61508 standard. They are certified by a well known certification organization, who review their documents and regularly sends an assessor to inspect their process and how they work. This certification is a very important sales argument and is a demanded from authorities, so it is critical to live up to it.

2.2. The Definition of the Impact Analysis

Since there is no common definition of Impact Analysis was found, we will present a couple of definitions from the literature. With help of that information we build a common definition that will be used during this thesis. Thereafter the Impact Analysis questions at the company are studied, the questions studied are presented in Appendix A. The description of questions that is presented in this subchapters is our understanding and interpretation of the Impact Analysis form. In the end of this subchapters conclusions

about what functions the Impact Analysis questions perform at the site, in relation to the definition is discussed.

The definition of Impact Analysis by Bohner and Arnold[1] is “identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change”. Impact Analysis comes from the need to estimate the size of the change and plan for the implementation. Another need is visibility, so that all effects are accounted for and unwanted side effects are contained. The Impact Analysis should help to understand which software artifacts are of need to be updated. Those artifacts could be: requirements, source code, design specifications, test specifications and other documentation.

From SWEBOOK[2] the main objectives of Impact Analysis are:

- Determination of the scope of a change in order to plan and implement work
- Development of accurate estimates of resources needed to perform the work
- Analysis of the cost/benefits of the requested change
- Communication to others of the complexity of a given change.

From Queille and Voidrot[3] the definitions of Impact Analysis is following: “ The task of assessing the effects of making a set of changes to a software system.” The goals of Impact Analysis are to minimize unexpected effects of the change on the system and make the change cost-effective.

The common definition created from information in the literature would be that it is important to gasp the size and cost of the change. The information from Impact Analysis should be used during the decision process and help decide, if it’s worth to implement the change. Only Bohner and Arnold in a explicit way mentions the importance of documentation during Impact Analysis process.

When developing safety products according to the IEC 61508 standard, analysis and documentation of the change is a part of the certification criteria. At the company this have been solved by adapting a Impact Analysis form that have to be answered by developers on request by CCB. Since Impact Analysis was first introduced, it has evolved from couple of questions to today’s thirteen questions, see Appendix A. Those questions have been created with interest of different stakeholders: CCB, Safety, Testers and Developers.

At the site, raw text documents are used to contain the Impact Analyses. The document is created by copying a skeleton with all the questions from the company's internal website. The answers are written down after each question and then both the questions and answers are copied to one note in the current Tracker Case.

The Impact Analysis form and documents related to it were studied [Internal 1-3], to gain an understanding of how it worked. To gain information on which questions were important, informal interviews with the stakeholders were done. They conveyed their picture of Impact Analysis form and the questions in it. Below follows a summary of what was understood about the questions in the Impact Analysis form. For those not familiar with the questions it is recommended to view the questions in Appendix A while reading the summary.

1. The developer have to describe why or why not the change is safety critical. This is one of the most important question, it set the importance of the change request. If the answer is “Yes,..” special precautions have to be made.

2. In this question the developer have to write down the solution to the problem. In the evaluation step there could be couple of solutions, but in the end only one should remain. Not implemented solutions should be removed from the Impact Analysis Note text and added as a Engineering note to the Tracker case.

3. To answer this question developers have to investigate if the error can be found in other versions of the program. This is done because different version of the software are maintained and the error should be fixed both in old and new versions. The CCB will generate new Change Requests to correct the error in other versions.

4. Here the developer lists which files/modules are affected by the change. The first part is directly affected files, which are files where code is changed. The second is indirectly affected files, which are for example ones that use same resources as the changed files. It’s very important that developers state if the files are SIL or not, it determines if the Test Team and the Safety Team need to be concerned with the case. This question is used by developers to find tests that verify that the code is not broken by the changes to the files/modules, also this question documents what files/modules where altered. The indirect effected part of the question works as a manual impact analysis and can be closely linked to our common definition of Impact Analysis.

5. This is a question with many sub questions, which makes developer thinks about important rules that have to be followed and document specific effects on the important parts of the system.

6. This question is used to document which SW/HW Library Items are effected and often it is the same information as in question 4. The information is used to inform people outside the local office about the changes in the library files.

7. Here the developer should document which documents are effected and create links to new Change Requests where the documents are changed.

8. The developers documents which verification tests have to be executed or writes new tests that are needed to test the change. The functional verification is always performed by developer before the case can move further on in the change process. In SIL cases CCB checks if the Verification Note consists of successful logs for the required tests.

9. The developer should document which user documentation ,for example manuals, that is effected by the change and create a new case that will be the one describing that specific change.

10. When CCB plans the change they need a estimate for the amount of work the change will imply in hours. The developer should with help of his/her experience provide a estimate.

11. & 12. To find the real problem developers should write what was the root cause and how to avoid that it reoccurs. This is used in lessons learned after a project is complete, to improve efficiency of future project.

13. Here the developer should state which functions or requirements are effected, both directly and indirectly. The CCB validation team use this information to create test runs for the Test team.

Summarizing the usage of the Impact Analysis at the site following can be said, it is mostly used for: documenting, checking if things are in order and finding items that are affected. From the interviews with stakeholders the questions importance to the product release were identified. The most important question is flagging if the change is Safety Critical or not. After that if the error occurs in other versions of the product and that all effected modules/files are documented and reviewed. It is also important that the validation team can see which requirements that need retesting.

In contradiction to definition in literature, the Impact Analysis at the company is more focused on the reliability and documentation, rather than on cost of making the change. Because of the safety requirements that exists, when something is broken it have to be fixed the cost is less important. The Impact Analysis at the site has to solve a wider range of tasks, then intended by common definition where it is mostly about seeing the effects of and planning for the change. Here the Impact Analysis holds a checklist of things that developers have to think about, it documents what solution is implemented and narrows down the amount of tests that have to be run. The conclusion is that it's made this way to provide control over the change during the whole change process and not just during the initial steps.

2.3. Description of the Change Process

To be able to understand the Change Process, two definitions from the literature are presented. They will serve as a common definition for those not familiar with the definition of the Change Process. After that the Change Process at the company is presented, this is done by analysis of documentations that are related to that subject at the site. The result is a detailed description of the Change Process and is unique in its completeness. In the end of the chapter the result is compared to the literature and conclusions are presented.

Software change process involves understanding the software change, specifying and designing the software change, implementing the change, and retesting the affected software. It is an iterative process where change specification can change when new ripple effects come to surface[4].

Change process from Daniels[5] consists of:

- motivating the need for a change
- understanding the change and documenting the change proposal
- reviewing the proposal and communicating about a need for a impact assessment
- creating the assessment and making a decision about the change
- documenting the decision and implementing it
- verifying implementation

The Impact Analysis at the company is just one of the notes in a Tracker case, with its 13 questions mentioned in 2.2, the change process as a whole holds other parts. They are summarized below and the process as a whole can be viewed in figure 1. This is a

description of the change process that should be common for all safety products at the site. It is compiled from the company's internal documentation: How To Handle CR in a Safety Tracker Project, CCB Handling in Safety Projects, Tracker Checklist, Deviation Handling for Safety Products, Configuration Management Process [Internal 1,2,4-6]. The description below is our interpretation of those documents.

A *New CR* can be created by anyone with an account in the Tracker software. Normally someone with knowledge of the error and the system will create the request, which is important because a good description of what is wrong is very helpful when assessing severity of the change. The CR is filled in according to documentation [Internal 1] and sent to the responsible CCB.

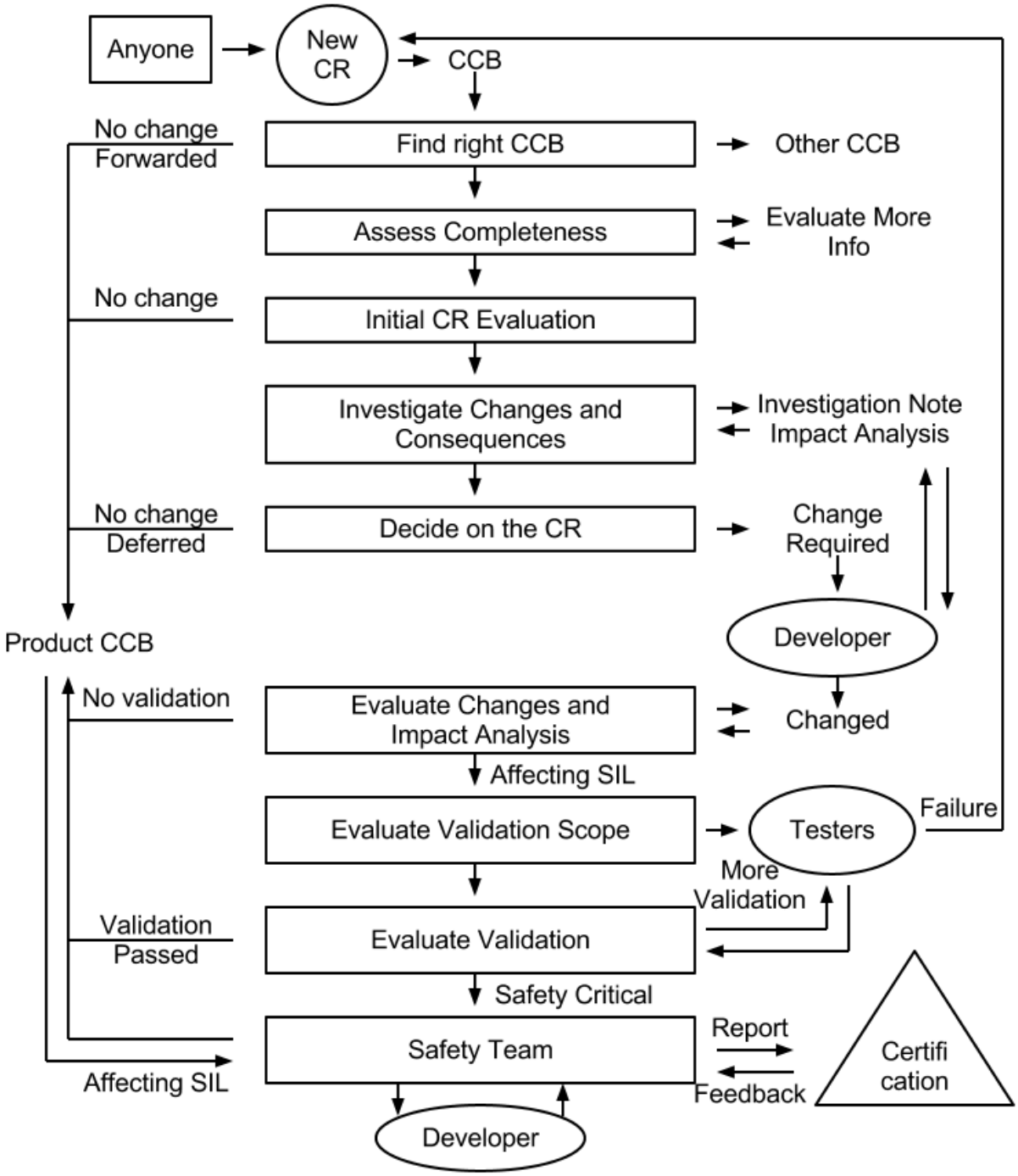


Figure 1 - The Change Request Process

If the CCB is not the right one, they are responsible of *Finding the right CCB* which have the responsibility for that component or decide that the CR have to be forwarded to a another product. In both cases notes about decisions should be added to the Tracker case.

Next the CCB have to *Assess Completeness* of the CR. If information is missing a note of what is missing should be added and the CR returned to its' submitter. When the initial criteria are satisfied, the initial CR evaluation can start.

At *Initial CR Evaluation* the CCB has to determinate the type of the CR, which could be: a new change, duplicate of another CR, a document change or a change due to ongoing development. The case that is studied here is when it is a new change, the other changes have shorter process description and not as formal. It is important to determinate if the CR should be done and move on to next step which is to investigate the change and consequences that can follow.

During the *Investigate Change and Consequences* step CCB have to make an Investigation Note and if the CR concerns safety classified product, an Impact Analysis have to be performed. In the case of CR being an old differed CR with an existing Impact Analysis, it has to be rewritten. When an Investigation Note is written a fast Impact Analysis is done and only some questions have to be answered. Otherwise the developers have to answer all the Impact Analysis questions and send the case back to the CCB.

Now in *Decide on the CR* step the CCB have more information about the consequences of the change and can make decisions based on that information. They should check that the Impact Analysis is done properly and decide if the change should be deferred or implemented. CCB should also confirm if any requirements are affected by the change, new requirement cases should be issued to handle that change before the original CR can be approved. Implementation should not happen before the requirements have changed.

Next step is the implementation of the change by developer, where he/she should update the Impact Analysis with the latest information and send the case back to the CCB. The role of developer is to implement the change, document the change and update relevant documentation.

Now in the *Evaluate Changes and Impact Analysis* step, CCB should evaluate the change and approve that the Impact Analysis is done right. When the Impact Analysis form is filled in special rules have to be followed, for example, no questions can be answered with just Yes/No. In those cases when the information is insufficient, cases have to be sent back to the developer. If the change was affecting SIL further steps are required before the case can be sent to Product CCB for closing. Also the CCB have to check that the Verification note is attached and the results are successful.

During the *Evaluate Validation Scope* step CCB Validation evaluates which tests need to be run in Product Type Test (PTT) by the testers, from the results of the tests a Validation note is created. If a test fails Change Request with the failed test is created.

The CCB evaluate the note in *Evaluate Validation* step, if the tests where satisfying the case can be send to the Product CCB for closing else more test runs is requested. When

a Tracker case is Safety Critical it always have to be sent to Safety team for a review before closing.

When a Tracker Case is being closed by the *Product CCB*, the case is checked using a checklist [Internal 4] for the last time before it is regarded as completed.

As mentioned in 2.1 the product at the company goes through a certification by the certification organization. This means that before a release can be done to the customers, a report have to be generated by the *Safety Team* for all cases that affect SIL and sent to the certification organization. All Tracker Cases affecting SIL are reviewed by Safety Team and reopened if something needs to be changed. The certification organization will certify the new product or give feedback on things that have to be approved before certification can be completed.

The change process at the site is very well defined and complies well with what is mentioned in the literature[4,5] , i.e. it will: investigate the change, perform Impact Analysis, implement the change in all related files and documents, validate the change. The Impact Analysis plays a big role in this process and could delay a release if not done properly. Maintenance of a complex product with many versions, requires the existence of a well defined process. For standard software development this change process could be too vast, but in this companies case it is what makes the product attractive to the customers.

2.4. Scope

The suggested scope from the company was to create a tool, but we felt that it was to narrow. From our academic studies we have learned that there were other important things that should not be neglected. The dynamics of people working at a company can be thought of as three interacting parts, the employees, the process adhered to and the tools that supports it. This can be depict as a triangle with the corners People, Process and Tools see figure 2 (Thanks to Lars Bendix for this idea and visualization). During this thesis for both the problems and the solutions we try to find which corner of the triangle they fit at. It is of interest to have solutions that can covers as much of the triangle as possible, which should help to create a solution that will perform even better then when only one corner is concerns.

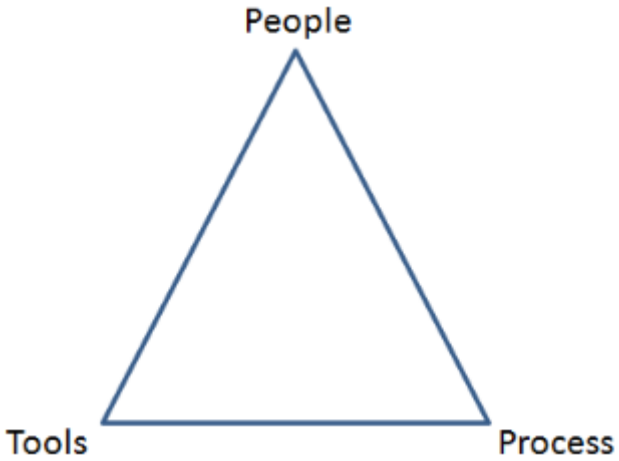


Figure 2 - The people, process and tools triangle

Some corners of triangle are more important than others. According to Leon [6], tools is the least important corner of the triangle. Also written by Lars Bendix and Otto Vinter [7] “What the developers needed was education in the concepts and principles of the CM

process instead of training in using the tool". Which means that tools are not as important as knowledge and understanding of the process. However after talking to couple of configuration managers on a Configuration Management Coffee Meeting, it became clear that a tool is needed as first simple step to create infrastructure to future process enhancements. First after a tool is made, a try do any changes in the process and education should be done. The tool is something everyone can see and touch, it will reveal some other problems and build confidence for new changes in process.

2.5. Developing the Tool

When developing the tool the XP methodology was followed [8], because it a good way of working in a small team. It is a methodology that we knew from school and it seemed as appropriate way of work to deliver a program on time and with high quality. That means that the code would be well tested and easy to maintain. For the most part pair programming would be done when working with the code. Every week we would let the customer decide on stories to implement. After a while we got used to coding in the new environment and language, so we could start working a bit faster and possibility to add a lot of features presented itself.

About this time we started to look for more customers, to widen the use of our tool to more than developers. This gave us many new XP user stories that could be implemented. Around this time we start thinking that we should even do some research for other solutions than tools. This presented us with a dilemma, there were a lot to implement at same time we wanted to spend less time on the tool. A solution to this was to implement the stores that we believed would be most giving for the company and to manage this we skipped pair programming and designing test to cover all the code. This way a lot of features were added and feedback could be gathered, that led to more accurate requirements from the stakeholders on the functionality that got implemented at the end. This way we can say that we really did a prototype and if the program is going to be used as a tool at the site it should be rewritten, specially the test coverage have to be better.

2.6. Limitations

In this thesis we limit our work to the latest version of the Impact Analysis for error corrections at the company (rev. E). The tool we made can also handle a earlier version because of their similarity.

The Tracker cases that we have viewed and studied in the Tracker system are error corrections (PRC) or enhancements. Our tool filters out other cases when looking for cases related to a specific release in the Tracker system, because the amount of cases shown would be very large and very few of them would hold an Impact Analysis note.

Because off time limitations, no metrics on how well the created tool is performing were gathered. This affects the lack of validation for how effective the created tool is.

3. Problem Analysis

In this chapter our goal is to find and analyze the problems that occur during the Change Process at the site, with focus on Impact Analysis. Most of the data in the analysis was collected in an informal way by talking to the employees and making unstructured interviews. First subchapter present the problem as it was presented by the company. Next follows a walkthrough of the change process to find the root of the problems. After which a subchapter shows specific problems regarding the Impact Analysis questions as they are today, this part is intended for people with access to the company intranet. Finally problems about loss of information are gathered in the last subchapter.

3.1. Problem Introduction

Now that the reader have better understanding of the context after reading chapter 2, it should be possible to understand the problems presented. It was stated in the master thesis proposal by the company, that the error correction process was not efficient and that the quality of error corrections needed to be enhanced. The presented problem from the sites management was that the error correction cases were sent back to the developers late in the maintenance life cycle. The errors were found during internal review of the Impact Analysis by Safety team and CCB, or in some cases external review of the Impact Analysis report by certification organization. As was described in Background an unsuccessful review by certification organization could stop a product from getting certified. The main concern was that the reviews of the Impact Analysis note was done too late in the error correction process (see 2.3) and that errors at this late stage were expensive to correct. The company proposed that this should be solved by making a tool that would help the developers to write the Impact Analysis correctly. Our approach to the problem as mentioned in 2.4 is to find alternative solutions and to dig deeper. This is done with help of literature studies, companies internal documents and interviews with different stakeholders at the site about the problems.

3.2. Root Causes Analysis

To find the real problems a technique called root cause analysis is used and is described in [9]. This is done to understand how the process is implemented and also to find and isolate the different problems. The principle is to ask a question and follow up with another question, until the root causes to the problem is found. The following subchapters will guide the reader through how this analysis was done.

3.2.1 Product CCB closing a change request case

The analysis started at the end of a Change Requests life cycle, which is when it closed by the Product CCB (see figure 1). This idea of starting at the end of the process was used because here we could find what errors that remained after the process steps (see 2.3) have been executed. The goal was to find out what errors where found at each step and when they were introduced. For some cases this is not the last step, for SIL the Impact Analysis report is done, this is discussed in 3.4.

Through structured talk with experienced employee who has closed many cases while working as Product CCB member, common errors were identified in the Impact Analysis note: questions were missing answers, had unacceptable answers like Yes/No without further motivation, missing notification about SIL levels on files/modules and even inconsistency with the information in the Tracker Case fields was noticed. For example missing SIL levels might indicate that the effect of the change on a SIL module/file have been overlooked. This also means that proper validation was not made and that the Tracker Case have to go back to a developer who need to validate that SIL-levels are written right and that appropriate tests will be executed. This lead to a lot of extra work,

because the implementation could have been done several months ago and the developer does not have the right build environment (product configuration) at hand any more. A nightmare scenario is when the patch is about to be released and a change request with missing SIL levels are found, which could delay the release to the customer severely. The question asked now and answered in the end of this chapter is.

- "Why does so many errors remain in the Impact Analysis at the end of the change process?"

3.2.2 CCB Evaluate Change and Impact Analysis

Looking back at chapter 2.3, where the process described how everything should work, it is easy to conclude that the faults should not exists at the stage when Product CCB is involved. All errors should have been dealt with earlier. By moving backwards in the process from the Product CCB to the CCB step Evaluate Change and Impact Analysis (see figure 1), new problems are found and new question is asked.

- " Why does the CCB fail to catch the simple errors?"

The process states that CCB has to check the Impact Analysis note and fill in the field in the Tracker system that states that the Impact Analysis is complete. The review of the Impact Analysis note is a manual process and errors when doing this can be classified as Human Errors [10], something done by a human. Human Errors can be split in two types, the unintentional error and the intentional error. The list below shows what kind of causes can exist at this stage in the process and answer the question above. This list has been created with help of gathered information from interviews with the stakeholders and evaluated with help of knowledge gained from the literature.

Unintentional errors :

- Occur during a long review session, because people can get tired from long interval with same task.[10]

- Factor like stress.[10]

- Because of the reviewer finding the task tedious, therefore performing less accurate. This could be because of the work is monotonous and is not challenging or stimulating. The employee feels that this task could be partly automated.[11]

- Also the reviewer finds the task unrewarding, because it lacks a sense of personal or collective benefit. This can lead to less careful check of the Impact Analysis.[11]

Intentional errors:

- For example the reviewer thinks his actions are correct or better then the action that is stated by the process. It could be that reviewers actions can lead to less obvious errors later in the process. While at that point it could be less obvious why a certain process had to be followed. This means that there are ways to bypass the process without getting any hints on the results of the preformed actions.[10]

- CCB have many other duties, which leads to prioritizing down the amount of time to review the Impact Analysis carefully. They intentionally choose to use the time for other tasks.

3.2.3 Developer implementing the change and updating the Impact Analysis note

Now it is not the CCB who writes the Impact Analysis answers, the ones who introduce errors in first place are developers. The process states that the developer who is assigned a change request first writes Impact Analysis and then updates it when an error correction is complete, to account for the ripple effects that have appeared from the first time the Impact Analysis was written. This is done before the Change Request goes to the Evaluate Changes and Impact Analysis state (see figure 1). The question we want to find an answer to now is.

- "Why do developers fail to find the simple errors when updating the Impact Analysis?"

The answer to this question from developers was in many cases that they felt that the first time they answered Impact Analysis questions during the Investigate Changes and Consequences step was correct enough or they just had forgotten to update the Impact Analysis note afterwards. The findings are that there is no system in place to remind the developers about the need of update or give them any feedback on common errors they have made. It was also noticed that the process documentation in many places draws attention to importance of doing the update of Impact Analysis, if new information is available, this seems to have been missed out by many.

Another observation in that stage is, for example, that there is no automatic system to keep the record of what has been altered. Which means that when updating an answer to a question in Impact Analysis with a new answer. The old answer is lost, unless it is saved by the developer in separate note. This means that there is no traceability to how the current Impact Analysis note has been created, which could be important if another developer need to understand how the note was created.

3.2.4 CCB Deciding on the change request

The first review of Impact Analysis is done when decision about the change request is taken by the CCB (see figure 1). This is the first time CCB see the Impact Analysis note. We ask the same question as before.

- " Why does the CCB fail to catch the simple errors?"

The causes are the same as in 3.2.2, but there are additional causes to why errors slip by. From interviews it has been evaluated that the information that is needed at this stage in the process, to make a decision, are questions 1 and 3 from the Impact Analysis note (see Appendix A). The other questions are not that interesting at the time and occasionally not reviewed fully. This happens because CCB don't have time to look at all questions, also they have a good knowledge of the system and can with little information decide the importance of the error correction. Another aspect is that all changes that are sent to the local development team have gone through multiple levels of support, which means that they are important to implement. Still some errors are more important than others and that has to be prioritized.

3.2.5 Developer writing the Impact Analysis note

Moving further down the process to the Investigate Changes and Consequences step (see figure 1), the root state where Impact Analysis is written by developers is found. By doing interviews where developers had to answers questions about Impact Analysis and explain the work flow, we were looking for answer to the following question.

- " Why does a developer fail to write the Impact Analysis correctly?"

The causes in 3.2.2 can be related to developers as well, if review is changed to writing the Impact Analysis. As developers and CCB are both doing manual work, many causes correlate. New findings from asking the question above were to developers are.

- The traceability is not available in the complex net of requirements, architecture designs, implementations and tests**.** As stated by Bohner traceability is the stepping stone to be able to make the Impact Analysis possible.[4] This missing infrastructure could be the cause to why the Impact Analysis is so disliked by the developers, it is simply very hard to identify things in the system. Nevertheless there are tools that help with some part of the traceability, not everyone are aware of how to use them. During thesis work we did not have time to understand how everything was coupled together. That's why we can't go deeper in how the current traceability works. It is noted that the new architecture system Enterprise Architecture (EA), is well received by the developer because of the possibility to use the traceability in it.

- The Impact Analysis at the site as mentioned in 2.2 covers a wider range of task than the common definition. This makes the Impact Analysis something that takes a lot of time to complete, because too much is asked. It was found that there is no easy way to answer some of the questions and developers have to dig deep, before a good answer can be produced. In practice developers have to practically implement and test a solution on a local system before the Impact Analysis note can be filled out. This is done early in the change process, at the Investigate Changes and Consequences step.

- It was observed that some information is not used immediately. As stated earlier only some question where of immediate importance for the CCB. This means that some of the work in the Impact Analysis could have been done after the Investigate Changes and Consequences step.

- Another problem inspired by literature is that the information gathered is not used in visible way[12]. This means that some information put in the Impact Analysis, is never used by anyone or, if it is no one really care to evaluate the quality of it. This being a good cause of why low quality information is put under some questions. Developers think that no one will read it because it is not clearly understood who is the stakeholders of a question. It is specially observed from browsing Impact Analysis answer that non-SIL cases have lower quality in the answers. This is because of no reaction from CCB or Safety to the answers that are written in those cases, there is no reward for putting time into writing good answers.

Further up in the process (see figure 1, page 11) no Impact Analysis exists and therefore no more analysis is done.

Too answer the question asked at the beginning of this chapter, we asked more specific questions and dug deeper below the surface. This resulted in problems that could be linked to all of the corners of our triangle philosophy. More specifically: the process is gathering too much information at the wrong time, the people lack motivation to work hard with Impact Analysis, there should be tools build into the environment to support the process and people.

3.3 Some specific problems related to Impact Analysis questions

This chapter pinpoints some of the observations and opinions directly connected to the Impact Analysis questions. The problematic questions are described in this chapter. Solving some of those problems is a fast way to improve the Impact Analysis process at the site.

Observations were made about poor usage of the information gathered by the Impact Analysis questions 10,11 and 12. Those questions are there for a good reason (see 2.2) and can deliver useful information. It's clearly understood by many employees that they must be in the Impact Analysis and can't be removed. At same time it is very demotivating for them to answer the questions, because there are few times when the answers are used for anything good. The questions are not used because of lack of time to handle and evaluate the gathered information.

Question number 2 have been pointed out as a problem by many interviewees, because it is hard to have a good answer to it without implementing the change on a local system. When this is done the developers test if the solution works with existing unit/design tests or write new ones if the change is something that have not been tested earlier. This results in that the Investigate Changes and Consequences step in the change process takes very long time, while the implementation step can be done very fast because many things are ready to be committed from the local repository to the public one (see figure 3, page 32). Another problem that was noticed is that other solutions that developers could propose in the Investigate Changes and Consequences step are lost at the Evaluate Changes and Impact Analysis step. Often the process to copy the other possibilities to another note is forgotten, which means that if in the future someone would not be able to find what other options about the change where proposed. Loss of this information easily outweighs the cost for space to keep it. The cost for personnel to retrieve it from scratch, is a lot higher than making a proper system that retrieves it at almost no cost.

The guidelines [Internal 3] on question 4b are long and hard to understand. The term "Indirectly Affected" is hard for many developers to gasp. Better information on how to handle this question should be in place.

In question 5i developers do not know if they should or should not write anything when the answer is "not applicable" (NA).

Question 10 should have better examples in the guideline for how to answer this question so that more useful information can be gathered, for example the answer should state the time to, analyze the error, implement the code and verify the change with tests.

Question 11 and 12 are often answered with standard answers that provide no new information. Also sometimes good answers are written, by the developers. This information is not used as it should be. The teams are too busy with new projects to use those questions and make lessons learned from them.

Question 13 is placed too late in Impact Analysis form. It is more important that questions 9 - 12, because it is always used and narrows down the amount of tests that need to be run by the test team. While question 9 - 12 are seldom used and recognized as less important questions by the stakeholders.

3.4. Information management

It was also observed that some important decision making and communication is unofficial, i.e. outside the Tracker. This happens often when a change request is submitted by an important customer (another system used for those errors, a support team makes them available in Tracker after a while) and have to be finished soon. Because of the pressure from the customer some levels of support are skipped. This means that there is no way to follow up and trace the decisions that were made, all because the process does not allow to push critical errors fast enough to the top level of support. Sometimes there is a need of a more effective way to get things done.

As mentioned in 3.2.1 the Safety Team find faults as well. Information about faults they have found is gathered too, with help of informal interviews and viewing the reports from review meetings. In those repots all found faults were summarized. This was mostly the same as for when CCB closed the cases, expect on one detail that is Safety Team specific. They have to make the Impact Analysis "look good", meaning that if the answers are to technical or in other ways not suitable to be included in the report to certification organization. It's the Safety Teams responsibility to catch answers that could make certification organization give feedback that will be too costly to implement and that would delay the product release. When an Impact Analysis with an error is found it will be reopened and sent back to the engineer who have to rewrite the unfitting answers. Observations where made that in some rare cases the original answer are richer on information and is better suited as an answer from a developers point of view. The problem here is that there are two different points of view on what suitable information is, the developers' and the Safety Teams. They use information in different ways. Developers need to see the details of the problem, while Safety need to secure that the answer will be understood by the certification organization. A conclusion is that it is bad if the information in the Impact Analysis note after the change is not as useful from developer point of view as before. This could cause more time spent, at a later stage, to determine cause of a new problem related to the original bug fix.

4. General problems and solutions

This chapter describes general problems and solutions, removing the context of the company. This is done for the sake of generalizing the thesis work so that it can be applied to more than the site. However many of the problems are inspired from the analysis of the site in chapter 3. The scope is that the problem should be repeatable in other large software development companies. In this chapter we try to follow the methodology in 2.4 and cover as many angles as possible. First the process is presented, with angle to how find the continues improvement path. After that the tool is presented where we describe what tools are good at doing. Last we cover the people aspects.

4.1. Process

Having a working process in place is vital for any company. In this subchapter we discuss how the process has to evolve together with the working environment. It is often observed that a legacy process in place is obsolete, which means it needs to be fully or partly reconstructed to fit into the current reality. The problems are:

- How do we find if the process is poor?
- How do we find a new process?
- How do we introduce the new process in a correct way?
- How do we confirm that the new process is better?
- How do we avoid obsolete processes?

To answer the first question, there is need of new eyes to identify if the process is not followed or if it is followed but it have flaws that need to be fixed. Often people working with a process are so worked into it, that they don't see the obvious shortcomings of it. In some cases they do, but they often accept it as unsolvable and just try to work around it. There is a need of someone with energy to push the change. Another way of finding if the process is insufficient is by looking at metrics and have a set goal for the values. Even then there is need for someone gather and calculate them. The discussion about how metrics and goals are set is outside the scope of this thesis.

When creating a new process the old one should have been studied, as there could be parts that can be reused. As a basis for improvements, to improve processes and design new ones, you must first obtain concise, accurate, and meaningful information about existing processes.[13] The employees working with the process should be asked about their ideas and the new process that will take form. The process should be theoretically run in the environment to validate it's improvements compared to the old one. This could lead to more ideas about improvements to the process.

When introducing the new process one should, if possible, use the buildings stones of the old process. People are busy and thus the less new things introduced the happier they will be. The process should be introduced with help of the ones who work with it. They should be a part of it and feel that they are co-owners.[12] This way it should be easier for everyone to accept the new process and follow it.

The question about if the new process works is something that is easily forgotten. The new process is implemented and then the people responsible never verify if it really did do any good. In worst case the old process was better than the new one. First if the process was introduced in the way described earlier it should be better than the old one, but that is not an excuse for skipping the verification. We know of two ways to verify the success of the process. One is to, ask the ones working with it and try to get a

understanding of the situation. For example if the employees are mostly approving the new process and they are happy with the process after it have been run for a while, then it is a success. From those talks new ideas could arise and the process would evolve with the working environment as intended. Just doing this however will not hold with any serious management, that's why we need to have metrics. Gather the data from metrics to be able to show with figures the improvement that has been made. Using both methods together could lead to understanding that there are other factors that can affect the process. For example we could see that metrics rise to a bad level, while we knowing that the process works as it should. This can lead that we can find other root causes to why things look strange or wrong at some points. Discussion about metrics is intended as future work.

A answer to the last question could be formulated with help of another question "Is right approach used by the company to handle the process improvement? ". Will the company settle for a inactive approach and solve problems as they arise or use a active one and motivate employees to spend time on process improvement. For example one could have a set of hours every week that are used for what the employee find useful, used by Google. Another example could be to have a employee who is responsible to gather and manage process improvements. The conclusion is that companies have to manage the processes or they will become obsolete!

4.2. Tools

As mentioned before the tool is often the solution everyone want, but does it really solve the root cause problem. In this subchapter we discuss what a tool should do and when is it appropriate to have a tool as a solution and when it better to focus on the people or the process. Also we discuss the need of keeping information than can be useful and having traceability to assist the user in understating the situation. Last we discuss the balance of having the right amount of tools.

Goals of tools:

- save time
- secure the quality of repetitive tasks
- perform tedious tasks
- guide the user
- gather information
- filter information

In some situations a tool is not able to replace a human, for example, when some kind of intuition is required. What is known, is that machines are a lot better than human on repetitive tasks. One way to streamline a working environment is having a engineer analyzing what parts can be automated and replaced it with a machine. This is how the modern industrial society have been created. For companies who employ highly educated personnel, innovations and improvements have to be promoted. To remove the tedious tasks from the working process is part of developing a better business. The cost for a person doing something a computer can do is so much higher.

The environment we work in as software developers is often a tool, this should be: flexible, save time and guide the employees to do the right decision. If something is wrong it should give fast feedback on know pitfalls to the user. But without proper understanding of the tool or by not following the process, the tool is just another layer of added features that stop the human from doing what they are best at, thinking and reasoning. It is known from literature[6,7] that a solution have to cover more than one

corner of the triangle (2.4), to be as effective as possible. The tool can never be a standalone solution to the problem, it should be an important part of the whole solution. It's common to manage code in version control system and in some companies same thing is done with documents. With today's low cost on information storage, there are no problems with having a tool that documents and does version control of every decision that is made. If a tool can store information then it could to be used by other people in future. That can save a lot of time and money for a company. The cost for having someone spending 10% of their time on trying to figure out how the final result of a decision was created is high. Making a tool that helps them to do it at almost no cost is better, especially if there would be multiple users for the tool. The cost for such tool is an investment for the company, it should have a good pay-back time[14] which should satisfy the need of the management as well. Another problem is when there is too much information, having a good intuitive interface in the tool is important. It should help to sort out the relevant information, which is critical for efficiency of the employees work.

Is more better, or one tool is good but more tools are even better? There is a difficult balance in the amount of tools that should exist. For example having different tools doing the same things could be bad because there is no common way of work in the company and more tools have to be supported. While at same time it gives the freedom to use the tool employee prefers to work with. What tools should be used should be discussed on management level, together with the floor staff. In some areas using any tool should be allowed and other there should be well motivated restrictions. An ideal scenario is when every tool supports the process and the part of the process they are meant to be in. Tools should not overlap because there could be confusion on when a specific tool should be used or not. The management have to see if there are too many tools around for some parts of the process, for example, if more than two screens need to be used to fit all the tools for a certain process. Then there probably too many diffident tools, this should encourage for investment in enhancement of the process so that fewer tools should be used or a better tool should be created.

4.3. People

The people problems are often very hard to fully gasp and generalize. In the chapters below we try to generalize in what we see as the tree main people problem areas: communication, motivation and education.

4.3.1 Communication

This is probably the most obvious problem in all big companies. Fred Brooks[15] described software constructions as " an exercise in complex(human) interrelationships." The more people there are the larger amount of communication channels there are, also the amount of information that need to be managed grows. Communication could closely be linked to the need of creating documentation and have traceability for decisions that are being made. This subchapter also mentions how communication can be more efficient with help of tools.

The first thing that is important to identify when a new change request is reported is to find who have the knowledge of the system part where the fault have appeared. This should be done in a structured way, for example as a written attachment to the error case and the people who are affected would get a notification. They should be able to verify if they are knowledgeable in the area, by editing the note with yes or no after their name. New names could be added to the list continually until the case lands at the right person. This step is important because it will build up a network of people as stated in [16]. This information can be used in future to find the right person directly and save time.

It could even be used by new developers to identify who to talk to if they need help with some part of the system. For example, based on this information a tool could help to point out which team or person to contact if working in a specific area.

People have to communicate to get things done, at same time we get a feeling that nothing gets done in a large organization. This because of all meetings that staff has to attend and all the documentation that have to be written for every action in the process. All this time could be spend on development and making a good product instead. Before a meeting is scheduled one should ask, if there are other ways to communicate that could be more efficient in this specific situation. What is the purpose of the meeting and what cost versus benefits are there for the staff who will be invited. Thinking this way will improve the efficiency of the company. When it comes to documentation of action one should try to find tool to manage the information in a way mentioned in 4.2.

To communicate ideas about improvements a suggestion board could be used. If anyone have a suggestion they should pin it up. This board could be used during a scheduled improvement meeting to see what idea there are. Another way is to have a wiki or forum, where some important information can be displayed and communicated.

Another important thing is to communicate the meaning of the change in the process and the results of it. It is important that staff understands why the changes have been implemented else it can lead to attitude problems, because changes are annoying specialty without any known results.

4.3.2 Motivation

This aspect have become more important in now days software companies. With the need of having well motivated employees to be able to compete with the other businesses. A well motivated personnel gives the company great advantages.

From "9 Things That Motivate Employees"[17] some interesting points about motivation can be discussed. The need of making the managements idea stick to the employees and it's a usual thing to feel rebellious to a process you have been told to follow. This motivates why it important to follow what has been written about the process in 4.1, make the employee co-owner of the process. Another point from the same publication states the importance of not pointing fingers and criticizing the employees. It's better to talk about how the problem can be solved and make thing work. This way they will get motivated and hopefully do a better job next time.

Some employees will be more motivated if they have greater responsibility, for example, a module or part of the code. If they do a good job they should be rewarded. This can be done in two ways. Public reward or private, it depends on the situation but giving both is the best way. The public one will motivate the co-workers to try harder, and the private one will make the employ feel more appreciated.[11]

4.3.3 Education

The need of having the right knowledge among the employees is vital for any company. We believe that good old power point presentation don't function well is some situations. The consent of workshop as a alternative way of educating and communicating the message to the employees. The two approaches will be discussed in this subchapter.

Power point is a fast way to present and visualize a solution to large audience. It's a well known tool, that everyone has heard of or used. While workshops is something that is best practiced with a smaller audience and is a concept that is growing in popularity.

The purpose of a workshop is not to have the solution handed to the staff, it should be created by them with help of the leader of the workshop. This way we can achieve the co-ownership of the process that is desired in 4.1. The solution will be created together and all the steps will be understood and motivated, in contradiction to having a handed solution. Thus they will have a full understanding of the process and situation they are in. However if the goal is to convey the big picture of the process in the company, to a large amount of personnel power point would be more effective. Both approaches to education will only work well if led by a skilled person, who has good knowledge on how to work with people and communicate the message to the audience.

Education can also be done in other ways, for example, if someone have a good idea about how to find a piece information that need to be accessed often, but could be tricky to find. This could be written on a wiki or a forum, where other personnel could easily look it up. The knowledge between developers can also spread with practices like pair programming and code review sessions.

5. Recommended Solutions to the company

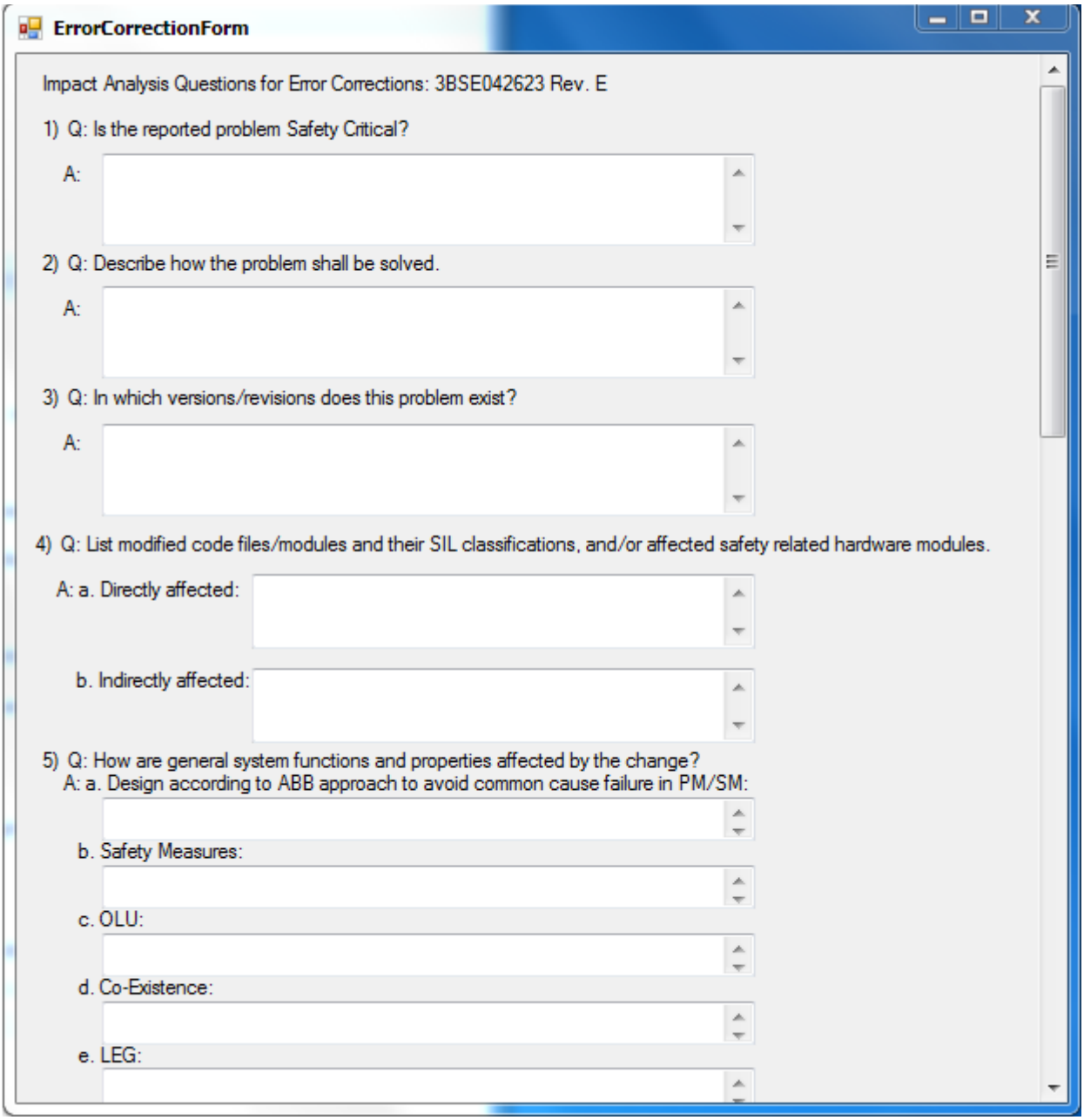
This chapter consist of the solutions that we recommend to the company. With subchapters covering: tools, process and people. In the last subchapter future work is presented. The presented solutions are the ones we believe are possible to implement in the known context of the site. For example it is know at the company that questions are hard to removed from Impact Analysis form, because of the 3-part certification organization The amount of work is so big that it removes the solution from being recommended to the company.

5.1. Tools

In this subchapter we write about the tool and how it will solve some of the problems at the site. Written in the steps we created our tool and what problems we wanted to solve in them. It also works as documentation of our work that can be used to inform of capabilities of the tool. In the end a discussion about the currently used tool (Tracker) for change request handling is presented.

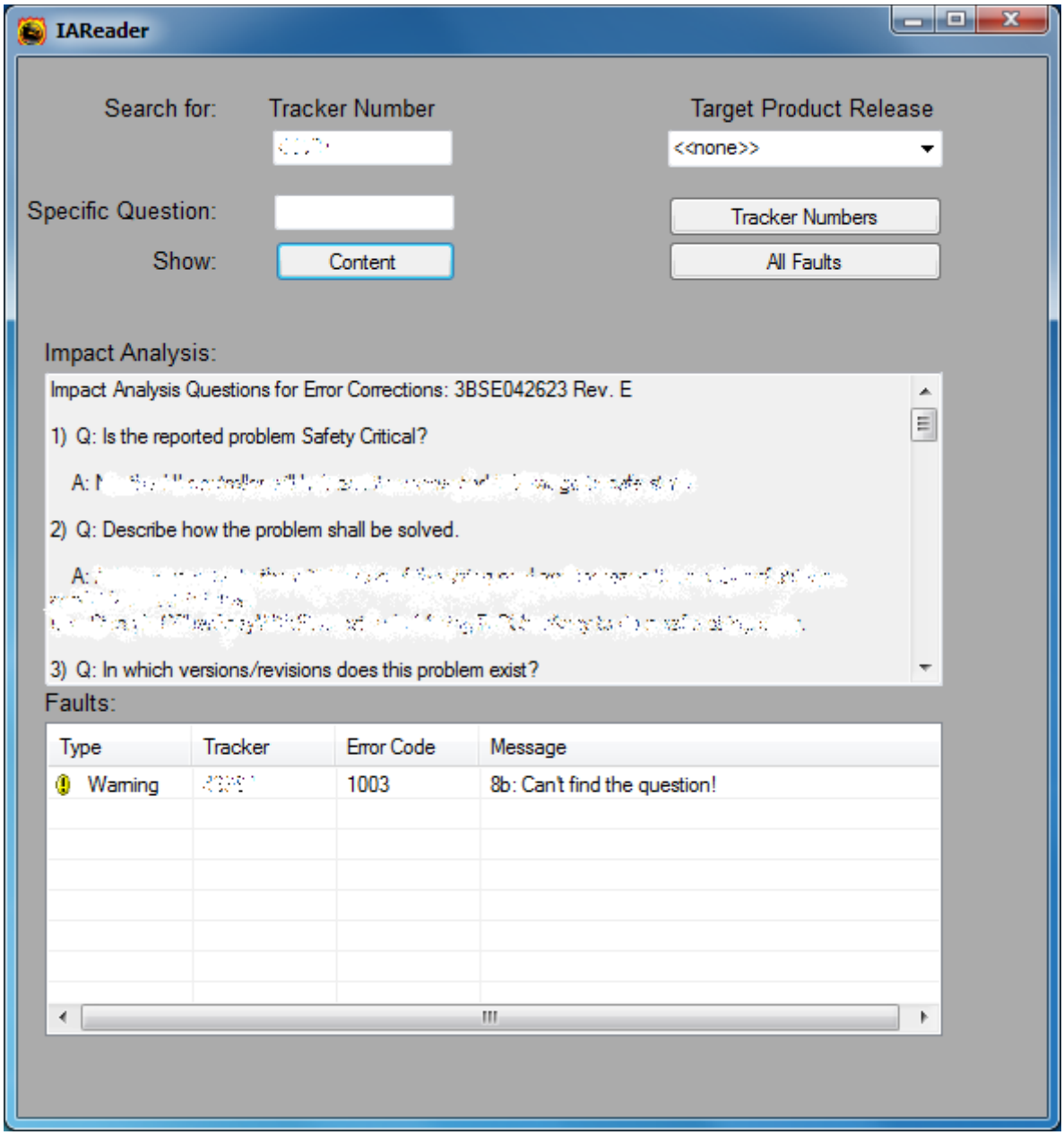
The problem we wanted to solve was that developers should write Impact Analysis without simple errors, for example, missing answer on a question, no motivation on a answer. The solution to this would be that we would create a static form (picture 1). But this solution received little attention at that moment because no one was interested in writing new Impact Analysis. What was more interesting is checking if the existing Impact Analysis note are correct. Which at same time provided us with opportunity to learn more about what kind of errors were common and to add warnings about them in the program.

The first prototype that was designed was a program that used a Tracker id (the change request case unique number) able to find the Impact Analysis note in the Tracker database. The note would be parsed by the program to find faults. This way it was easy to see if there was a fault of the type mentioned above in an Impact Analysis. The first prototype could be used by anyone to check if a specific Impact Analysis had some of the known errors. We also found out that there was many Impact Analysis form versions. As mentioned in limitations(2.6) only the latest two version could be parsed by our program. This was done to faster deliver a prototype, other versions can be added on a need basis. What we found more interesting is that the Impact Analysis form was edited by hand in some cases. This lead to many strange warnings where we could not find the end of a question or beginning of a answer. This stressed the idea of having a fixed question form (picture 1) so that parsing could be done easier.



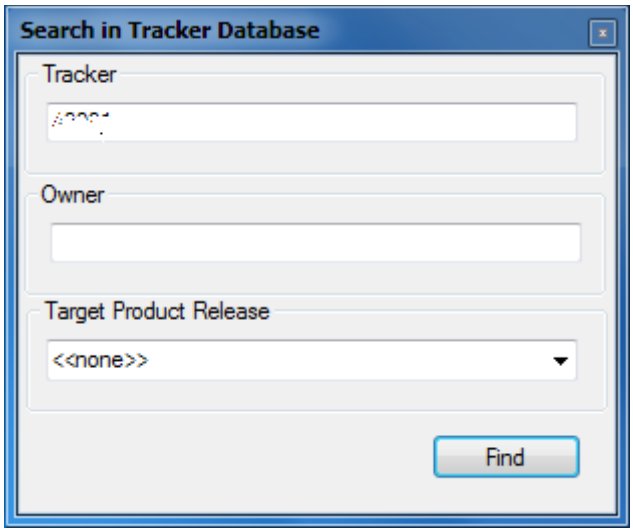
Picture 1 - The Impact Analysis form we created

The feedback from our first prototype was positive, but one feature was demanded before it would be useful. The demanded feature was to be able to search for a Target Product Release in the Tracker database and parse all cases that have an Impact Analysis. The new user interface looked something like this picture 2. This created a list holding all small errors for that specific release, the list was given to the Team leaders (have CCB roll often) so that they could mail the developers who had the simple errors our program had found. It was given to CCB and not directly to the developers, because CCB knew who to mail about a specific tracker case and we wanted to show our work to CCB as well. This feature was appreciated by CCB and Safety Team because they knew that next review would hold less errors of this kind. No specific reactions where registered from the developers.

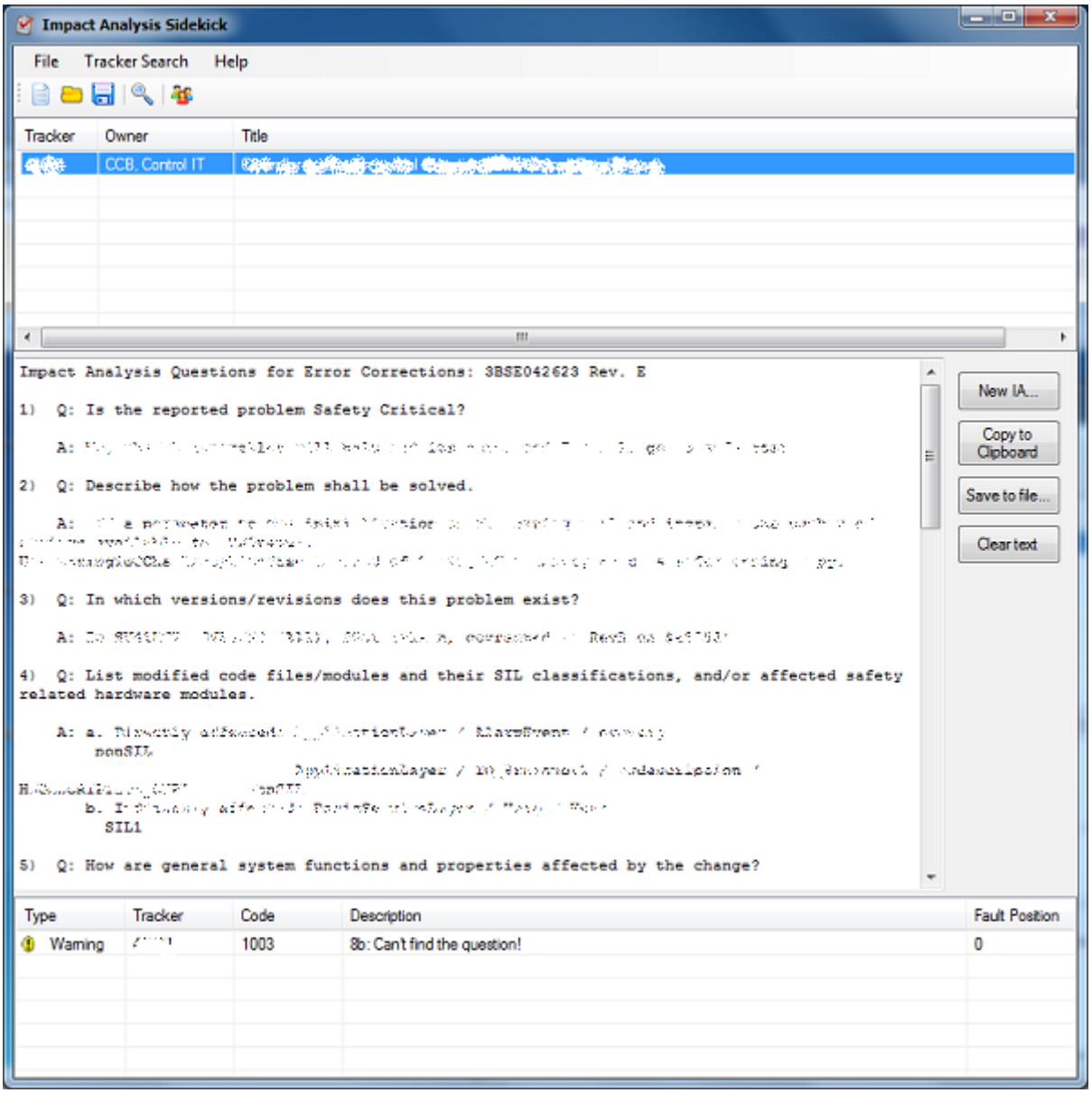


Picture 2 - The Prototype with Target Product Release option

Next step was to help developers to find if there were any errors in the cases they have written. So we added a feature where the program searched the database for all Impact Analyses connected to a developer name or id picture 3. After that we redesigned the interface, with new skills we gain in programming with C# from our first prototype. The new tool looked the following way see picture 4. Features to help developers even more were added, for example, when a fault is selected in the fault display, the tool would now show the Impact Analysis in editable window. Here the developer can edit the whole Impact Analysis note and see if any warnings remain. Then the text have to be copied to the Tracker tool and manually added to replace the old note. Also, the feature for writing the Impact Analysis from scratch in our tool was added, by pressing the New IA button the form in picture 1 shows up. This would create an easy to parse note and guarantee that all common faults would be shown. Still this would not solve all problems mentioned in chapter 3, but hopefully remove the need for CCB and Safety Team to return Tracker cases to developers because of simple errors.



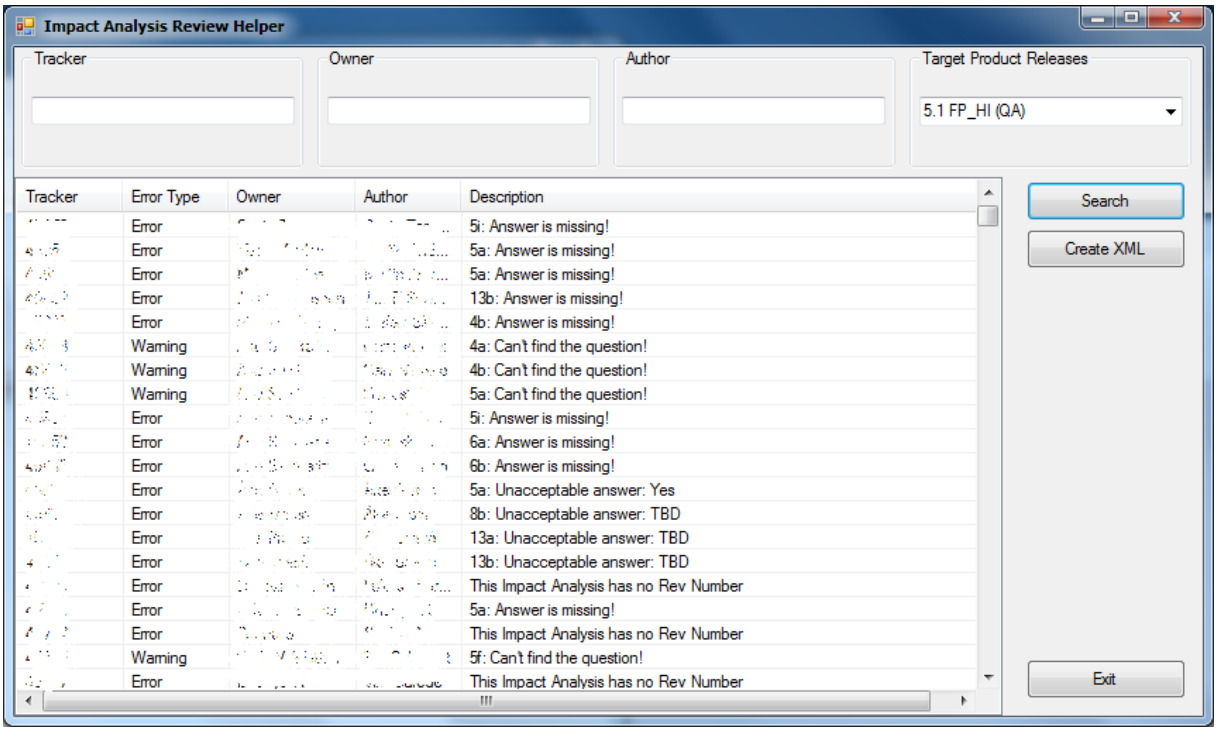
Picture 3 - The search window



Picture 4 - The developer view of the tool

After showing the tool to CCB we saw potentially that they needed other information than the developers. They did not need to see the Impact Analysis note, what was of bigger interest is to view the errors that the program had found. It was also of interest to know who was the author of the note. This lead to that a separate interface was created for CCB see picture 5, that would better satisfy their need of showing many faults and finding the author of the note. There is a possibility that errors remain, even if developers have our tool it is not sure if they will use it. That is the reason to why CCB need to check the cases with our tool as well. Hopefully this will free some time for them to read the answers instead for searching for simple errors. One useful feature that was added for CCB view is ability to create the list in XML. The file can be opened in Excel and there it is easy to filter the information after the users delight.

Because of the tool was developed near the intended users, as XP states[8], it should solve the problems that were presented. What is missing from the text above is a precise motivation of some actions and ideas. This is because we worked iteratively, if someone pitched an idea we would test it and keep it or remove it depending on how well it was received by the stakeholders. The process of when and why something was added was not documented, but we recall the main features that are mentioned above.



Picture 5 - CCB view.

There are still many ideas about features that where designed but not implemented, they will be mentioned in the Future Work subchapter.

The tool currently in use to handle the change request and the notes attached to them is Merant Tracker. In this tool the Impact Analysis or any other note is attached to the case and viewed. This is the place where decisions about the case are documented. What was observed is, that when a change to a note is made the only information available is that it was changed. There is no way to find the old note in the system, because it is gone forever. This problem has been mentioned in many places during the thesis. There

is no way to follow the actions that have been taken on the notes. Our recommendations are:

- Use of another software for change management and ticket tracking (change requests)
- Creating a new software that will work with Tracker and manage the notes, keep them under version control.

The big drawback of the first solution is that change the system is painful and finding a new software requires time. While there will be a need to do so in future, because Tracker is getting old. However some observations about a new software were made, it should have been implemented during 2011, but that was impossible to achieve. It's important that the company can secure that the new software has the right functionality before it's decided that it's time to use it. The second solution could be good for now, but in the end it will be lost when the new tool arrives. Also it will add another layer of features on the Tracker, this tool should be done with great care.

5.2. Process

In this subchapter we propose improvements to the existing process, which are inspired by Chinese saying about the three factors for success: right time, right place and right people. This solution is written with help of knowledge gained when in chapter 2, studies about Impact Analysis questions and the Change Process were done. The problems that are solved here have been found during the problem analysis in chapter 3. For example that some questions are hard to answer, when answering the questions the developer should be sure that the answer will be used by someone else and that his/hers time is not wasted.

One of the things about the Impact Analysis at the site mentioned in chapter 2.2 is that it covers a wider range of things than intended by our common definition of Impact Analysis. What was observed is the scenario in figure 3, this means that the time to make initial Impact Analysis note is vast. Also many developers implement a lot of code during this stage. It was observed that all information gathering is not needed for the decision about the implementation. There is a problem when a case is deferred, because a lot of the work put in the Impact Analysis is wasted. When the case is not deferred the total amount of time spend working with the change should be the same, for both the process today and the suggested improvement. What was noticed during the process studies in 2.3 that there is a way to create an Investigation Note where only some questions have to be answered. This means that this process is partly there, but is not used. The motivation for answering all questions at the initial Impact Analysis is shallow and the goal is just to gather as much information as possible. Also some question make developers write code. Evaluation of how information is gathered and used, has not been done by the company before.

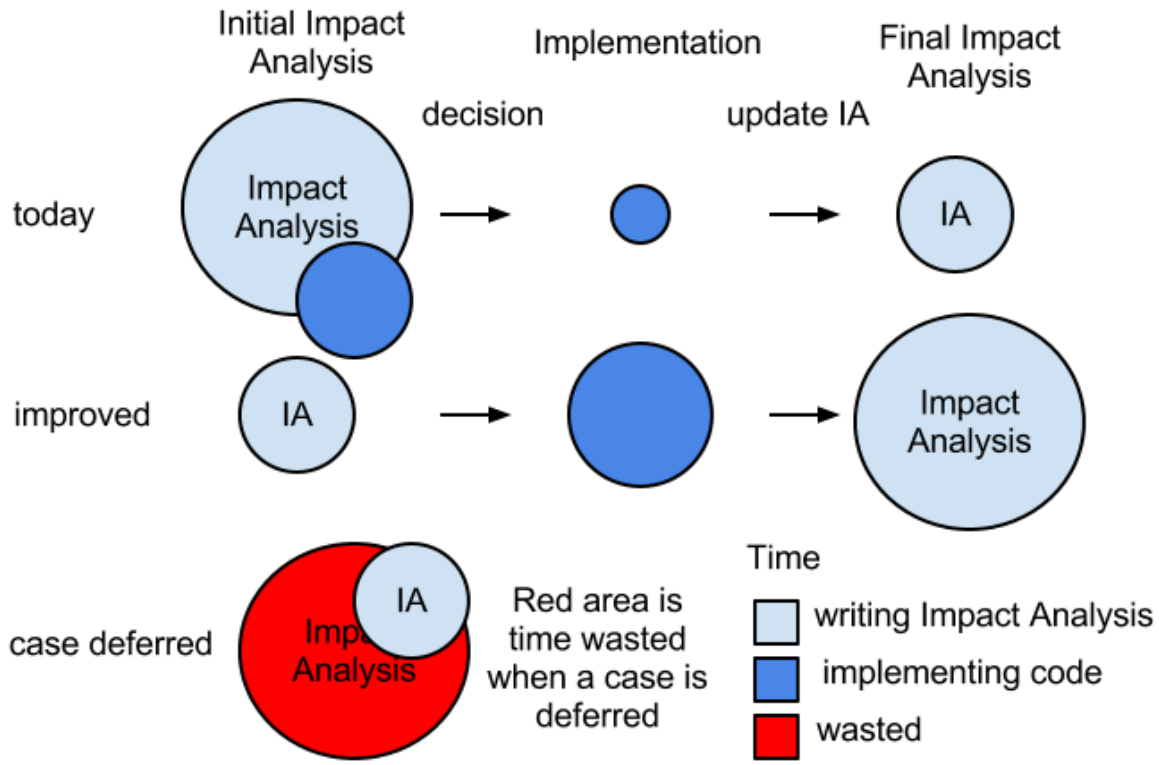


Figure 3 - The visualization of time spend in different stages of the process

From the problem analysis (chapter 3) it was learned that often question 1 and 3 from the Impact Analysis were enough for the CCB to be able to make the decision. Other questions could be useful when the change request case is hard to figure out. This leads to the following solution. The two questions used by CCB for decision making should be mandatory to answer. While other questions would only be answered by developer if it is required by the CCB or if the developer have a good answer to the question already, adding "zero" cost. This means that more focus will be placed on the Impact Analysis note after the decision is made. Other benefit of having it this way is that the developer can fill in the rest of the note when the change has been implemented and partly tested. This means that it will document better information about the change. A problem is thus that if a change requires a change to requirements documentation, that change has to be implemented and closed first, before the code can be changed. It now will be in the developers hands to contact the CCB to create the change request for the requirements documentation, implement the change in documentation, inform the CCB about it and first when it's closed get back to the implementation of the code.

5.3. People

Education and knowledge spreading should be done so that developers can learn the best ways to answer the Impact Analysis. Solutions mentioned in chapter 4.3 can be used by the company to fill in the people part of triangle. In this chapter we want to discuss how to find a solutions to problems and show that there is a need of combining all the corners of the triangle, with main focus on people. Also the timing of lessons learned in projects at the site is discussed.

The problem is that to answer all Impact Analysis questions the developer has to work near the code, this is especially true for questions 2 and 4 of the Impact Analysis (Appendix A). We believe this is the main problem of the Impact Analysis, because of the

amount work that need to be spent on them to get a good answer is too large. From literature we know that “The change impact analysis techniques need to be applied initially at the architecture design level to capture module dependencies without being dependent on coding style and coding techniques.”[18] This means we need a tool that can take care of dependencies and show the architecture in a usable way. But even if we know what parts are effected we can’t answer question 2, because this questions is really hard to answer without looking at the code. To this we see two solutions, the process have to be altered where there is no need of answering question 2 at the initial Impact Analysis if the answer is not obvious to the developer. The second one is to educate people in understating how things are working in the system so that a solution can be designed on a level where code does not have to be written. We want to avoid the scenario were a lot of work put in Impact Analysis and the code is discarded, this is pictured in figure 3. This problem can be solved with education, from interviews we know that there are developers who can find solutions without coding. Those developers should run a workshop on how to find solutions in more efficient ways. Overall the conclusion from the work that is done around change request is very like being a detective. This means that knowledge is the key and finding the right information effectively is the problem. If the information is there in the first place, the need for asking the right person for help arises here. The solution from chapter 4.3, a visible network of what people are used to work with could be very helpful in situations like this.

The deployment of skills mentioned above could be done, for example, with a workshop where a team sits together with a leader, who should be knowledgeable on solving the Impact Analysis mysteries, and write an Impact Analysis. This should not be done half-hearted, during this workshop a perfect Impact Analysis should be created that could work as reference for the future. During the workshop it is important to take in consideration that some teams work with SIL code and other only work with non-SIL. The management has to sell the benefit for people working with non-SIL for why they have to write good Impact Analyses. For example, it saves the company money when testing or it gives credibility on that everything in the product is done according to the high needs of the Safety Software standard. A faster way to spread the knowledge could be to educate one person in every team and then let them be ambassadors of the new way of work in their teams.

A way to solve problems with communication is to enforce communication opportunities between the people, specially between different stakeholders. For example, the process improvement meetings have to take place at right time point during a project. Management should allow time to be spent on this task, but in return demand some results. Currently a new project at the site starts before the last phase of the old one have ended, which leads to that no one spends any quality time on learning and improving the process. It might be better to evaluate lessons learned in the middle of a project.

5.4. Future work

There is still much work that can to be done to solve all problems at the company. There is especially much that can be done to improve the tool, for example, the tool could collaborate more with other tools at the site, save the Impact Analysis note in the Tracker database, etc.

An analysis of how many errors are found contra the amount of the Impact Analysis notes parsed should be done. This would point out how serious the problem in reality is and how much of the problem we can remove. The only metrics we have is that it took

about 800 man hours to review and "repair" all Impact Analyses notes for a release a couple of years ago, that is something the company want to avoid from happening again. We mention that metrics for process are of importance and should be studied further. The following paper could be a good start "Defining and Applying Metrics in the Context of Continuing Software Evolution"[19].

Most of the data in the analysis was collected in an informal way by talking to the employees and making unstructured interviews. One way is to make a solution from your own intuition and refine it by presenting it to as many persons as possible in an informal way. This technique is very efficient in getting feedback on the ideas and will come to a conclusion that will get a wide acceptance among the ones it has been presented to. One drawback is that it might be hard to present the data supporting the resulting solution. Carrying out formal interviews could assess the problem and solid proof for solutions would exist. This is something that could be done to improve this thesis credibility.

6. Conclusion

In this thesis three areas have been explored and we believe that they are fundamental for a successful business, they are: process, people and tools. By studying the Impact Analysis and the change process that exist at the company, deeper knowledge about the site have been gained. With help of a root cause analysis the initial problem was widen and deeper insight was achieved.

First we found out that the Impact Analysis is used more for documentation than economic concerns, which is the common reason for having it. At the site it is used for a much wider purpose and gives great control of the change. Also a complete picture of the change process was created, showing the whole process in one figure, see figure 1.

During the thesis a tool was developed that finds the errors in the Impact Analysis note. The development was done in a iterative fashion and new features were added as time progressed. The final conclusion from displaying the tool to the users is that it can save a lot of time and find many small errors in early stages of the change process. This should give the CCB, developers and Safety Team more time to spend on other tasks.

One of the ideas about improving efficiency of the change process is making the process more lightweight during the initial steps. The amount of work that needs to be done before the first decision can be made is too vast, which leads to long lead time before decisions can be made. There are two ways to improve the situation the company is facing today. The first is that right information should be written down at right time. Only the information that is actually needed for the evaluation should be gathered and rest can be gathered when the decision to implement the change have been taken. The second is that the developers should be educated in how to find answers without looking at the code or solving the problem before it is decided that it should be solved. There are employees that are better at finding solutions fast, this knowledge should be able to spread to other developers at the site.

Another problem found is that developers don't see the purpose of spending time on tasks that are waste of their time. The information gathered by the Impact Analysis has to be used so that employees have a reason to write good answers on the questions. There should also be better knowledge amongst developers why the process is the way it is, this can be done by educating them in a different way. That will involve them in the decision making when the process is created.

The communication of information can be enhanced with introduction of internal forums or wikis. This way knowledge can be gathered and used by everyone at the site.

Our final conclusion is that a tool is the gateway to a solution, but without proper support of process and education of people. It will not perform at its full potential, because there are always deeper problems below the surface. A good solution will cover all three areas and is the way to a successful software development.

Bibliography

1. Bohne Shawn A. r and Arnold Robert S., "An Introduction to Software Change Impact Analysis", IEEE, 1996

2. Abran A. et al. "Guide to the Software Engineering Body of Knowledge" (SWEBOK), ISO Technical Report ISO/IEC TR 19759. IEEE 2004. URL: http://www.swebok.org/

3. Queille, J.-P. Voidrot, J.-F., "The Impact Analysis Task in Software Maintenance: A Model and A Case Study", IEEE 1994, pp. 234 - 242

4. Bohner Shawn A., "Impact Analysis in Software Change Process: A Year 2000 Perspective", IEEE 1996, pp. 42 - 51

5. Daniels M. A., "Principles of Configuration Management", Advanced Applications Consultants, Inc., 1985 (chapter 2)

6. Leon A., "A guide to software configuration management", Artech House, Inc., 2000 (chapter 11)

7. Bendix Lars and Vinter Otto, "Configuration Management from a Developer's Perspective", EuroSTAR, 2001.

8. Kent Beck, "Extreme Programming Explained: Embrace Change", 2nd Edition, Addison-Wesley, 2004.

9. Duffy G. Moran J. and Riley W., "Solve the Real Problem Using Root Cause Analysis", *Root Cause Analysis* - ASQ - Quality Management Division.

10. Rooney J.J. Vanden Heuvel L.N. and Lorenzo D.K., "Reduce Human Error", Quality Progress, 2002, URL: http://www.asq.org/

11. Whitaker K., "Motivating and Keeping Software Developers", USDATA Corp., IEEE, 1997

12. Shaowen Qin, "Managing Process Change in Software Organizations: Experience and Reflection", Softw. Process Improve. Pract. Wiley InterScience 2007; 12: pp. 429-435

13. Wolf A. and Rosenblum. D., “A Study in Software Process Data Capture and Analysis“ Proc 2nd Int'l Conf. Software Process, IEEE CS Press, Los Alamitos. Calif., 1993, pp. 115-124

14. Skärvad P. and Olsson J., " FöretagsEkonomi 100", LIBER, Upplaga 12, 2006

15. Brooks F. " Mystical Man-Month", Addison-Wesley 1975

16. de Souza C.R.B. and Redmiles D.F., "An Empirical Study of Software Developers Management of Dependencies and Changes", ICSE'08, 2008, pp. 241-250

17. Pozin I., "9 Things That Motivate Employees", Inc.com, 2011, URL: http://www.inc.com/ilya-pozin/9-things-that-motivate-employees-more-than-money.html

18. Urjaswala Vora, "Change Impact Analysis and Software Evolution Specification for Continually Evolving Systems", IEEE, 2010, pp. 238-243

19. Ramil J.R. and Lehman M.M., "Defining and Applying Metrics in the Context of Continuing Software Evolution", IEEE, 2001, pp. 199-209

Internal documents available only from company's intranet:

1. How To Handle CR in a Safety Tracker Project, 3BSE029592.
2. CCB Handling in Safety Projects, 3BSE030309.
3. How to fill in the Safety Impact Analysis Questions**,** 3BSE058012.
4. Tracker Checklist Main Flow, 3BSE062644.
5. Deviation Handling for Safety Products, 3BSE028806.
6. Configuration Management Process, 3BSE033641.

Appendix A: The Impact Analysis Form

Impact Analysis Questions for Error Corrections: 3BSE042623 Rev. E

1) Q: Is the reported problem Safety Critical?

A:

2) Q: Describe how the problem shall be solved.

A:

3) Q: In which versions/revisions does this problem exist?

A:

4) Q: List modified code files/modules and their SIL classifications, and/or affected safety related hardware modules.

A: a. Directly affected:
b. Indirectly affected:

5) Q: How are general system functions and properties affected by the change?

A: a. Design according to the company's approach to avoid common cause failure in the products (*this question have been slightly altered*):
b. Safety Measures:
c. OLU:
d. Co-Existence:
e. LEG:
f. Performance and memory consumption:
g. Real time behavior:
h. Restart behavior:
i. Any other (if applicable):

6) Q: SW: Which Library Items are affected by the change?

A: a. 1131 Library Types:
b. 1131 Firmware functions:
c. HW types:
d. HW libraries:

7) Q: Which documents, ids & titles, need to be modified?
(e.g. PRS, Architecture, DoF, IFD, DD, Schematics, FTTD, LLITD, DTD, Parts Lists, FMEDA)

A:

8) Q: Which test cases need to be executed?
(e.g. DT, LLIT, FTT/CTT, Sequence tests, Environmental/EMC tests, FPGA simulation)

A: a. Directly affected:
b. Indirectly affected:

9) Q: Which user documents, including Online Help, need to be modified?

A:

10) Q: How long will it take to correct the problem, and verify the correction?

A:

11) Q: What is the root cause of this problem?

A:

12) Q: How could this problem been avoided?

A:

13) Q: Which requirements and functions need to be retested by PTT/SVT?

A: a. Directly affected:
b. Indirectly affected (based on architecture):