

Towards a Unified Process for Automated Traceability

Master of Science in Software Engineering Thesis

By:

Carlos Castro-Herrera, MBA

carloscastroh@ieee.org

Research advisor:

Jane Cleland-Huang, PhD

jhuang@cs.depaul.edu



Center for Requirements Engineering

School of CTI, DePaul University

Chicago, IL, USA

Spring 2007



Acknowledgements:

I would like to extend my utmost gratitude to Dr Jane Cleland-Huang for her help and invaluable input in this thesis. Her passion for knowledge, her hard work and her constant dedication have been an inspiration for me. I also gratefully acknowledge the feedback and input that I received from the thesis committee: Dr Orlena Gotel, Dr Xiaoping Jia, and Harold Streeter. Additionally I would like to extend a thank you to Brian Berenbach of Siemens Corporate Research for his early feedback and to Stephen Clark for his contribution of the trace strategy diagram shown in Appendix 2: Trace Strategy and Granularity.

The work described in this thesis was partially funded by NSF grant CCR- 0306303 and by a Fulbright grant.



Table of Contents

List of Figures	5
List of Tables	6
Abstract	7
1. Introduction	8
2. Traceability.....	10
2.1. Definition	10
2.2. Use	11
2.3. Importance.....	12
2.4. Implementation	13
3. Automated Traceability	16
3.1. Definition	16
3.2. Benefits and Limitations	18
3.3. Experimental Results	18
3.4. Best Practices for Automated Traceability	19
4. Process Groundwork.....	21
4.1. Benefits of a Process	21
4.2. The Automated Traceability Process Meta-Model	22
4.3. Paradigm for Creating Processes – Eclipse Process Framework.....	25
4.4. Content Starting Point – Basic Open Unified Process.....	27
5. Process for Automated Traceability.....	30
5.1. Building blocks – Method Content.....	30
5.1.1. New Method Content	30
5.1.2. Modified Method Content	33
5.2. Linking the Building Blocks – Process Content.....	36
5.3. Usage of the Sample Process Add-On.....	37
5.4. Modification of the Sample Process Add-On.....	38
6. Validation of the Work.....	40
6.1. Meta-Model Support for Traceability	42
6.2. Meta-Model Support of Best Practices.....	42
6.3. Process Add-On as Instance of Meta-Model	43
7. Conclusions and Further Work	45
8. References	47
Appendix 1: Automated Traceability Facilitator	49
Appendix 2: Trace Strategy and Granularity.....	50
Appendix 3: Traceability Request	52
Appendix 4: Traceability Results	53
Appendix 5: Create Trace Strategy	54
Appendix 6: Run Automated Traceability Analysis	56
Appendix 7: Set Up In-Place Traceability	58



Appendix 8: Test and Verify the Automated Traceability Results 59
Appendix 9: Automated Traceability 61
Appendix 10: Guidelines for Creating Traceable Documents..... 63



List of Figures

Figure 1. Sample Traceability Links	10
Figure 2: Sample Traceability Matrix in the Commercial Tool Caliber.....	14
Figure 3: Schematic of Automated Traceability.....	16
Figure 4: Experimental Results With Different Datasets Using Poirot.....	18
Figure 5: Benefits of Adopting a Formal Process	22
Figure 6: Automated Traceability Process Meta-Model.....	24
Figure 7: Bridging the Gap Between Process and Project	26
Figure 8: Separation of Method Content and Process Content [22]	27
Figure 9: Content Areas and Roles of the OUP/Basic	28
Figure 10: Lifecycle of the OUP/Basic	28
Figure 11: Linking the New Tasks Into the OUP/Basic Workflows.....	37
Figure 12: Mapping of Meta-Model to PDCA	42



List of Tables

Table 1: New Method Content	30
Table 2: Modified Method Content	34
Table 3: Mapping Between Meta-Model and Sample Process Add-On	43



Abstract

Automated Traceability presents a new way of implementing traceability that can potentially save time and effort over the traditional approaches that require links to be set up and maintained manually. However, in order to maximize the results of this technique, automated traceability has to be implemented within the context of a software engineering process. In this thesis we will present a generic process meta-model that will guide organizations in incorporating automated traceability into their own software engineering processes. We will also provide an example of an instantiation of this meta-model for a particular process and tool. This example, besides illustrating the instantiation of the meta-model, will present a paradigm and technique used to build processes and it will also serve as an open source content starting point.



1. Introduction

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.” [5]

The previous quote was written in 1987 by Dr. Frederick Brooks for an article titled “No Silver Bullet: Essence and Accidents of Software Engineering” published by Computer magazine. This quote is still as true today, twenty years later, as it was in 1987; and chances are that it will continue to hold true for many years to come.

In this quote Dr. Brooks alluded to the problems surrounding the requirements for a software system. For such a seemingly simple thing: what the system must do (functional requirements) and what qualities it must possess (non-functional requirements) [33]; there are plenty of issues that haunt it. Questions like: are they well understood?, are they all accounted for?, are they clear and correct?, are they well documented?, are all of them implemented in the system?, what happens if they change?, what will be the impact on the cost and schedule when they change?, among others are just some of the many questions that software engineers have to face on a daily basis when dealing with the complexities of the requirements.

One thing is clear though, the requirements are crucial to the success of any software engineering project. They are a key part of the documentation, they help converge the interests and understanding of the stakeholders [27], they are essential in managing the risks of a project in terms of the impact to cost and schedule [36], and they provide the initial input into the subsequent activities of design, implementation and testing. The CHAOS report, published by the Standish group, has consistently listed requirements related problems as a key failure factor in most of the impaired and challenged projects [35].

This thesis will address one specific problem that affects requirements, the problem of traceability. This work will try to demonstrate how a combination of techniques, tools and processes can help software engineers handle this problem and manage the requirements of a software system more efficiently and effectively.

The thesis starts by defining what traceability is in Chapter 2, listing its importance and common ways of implementing it. This is followed by a description in Chapter 3 of a technique called automated traceability that is used to operationalize it. Following this description, the pros and cons of this technique are listed along with some early results and identified best practices. Chapter 4 will explain why this technique should be used within the framework of a process in order to achieve better results.



This chapter starts by listing some of the benefits of having a formal process in place, and by explaining why the best practices for automated traceability should be part of a software engineering process. It then lays the groundwork for creating a new tailored process for automated traceability. It does this describing a generic process meta-model for automated traceability, the framework chosen to instantiate this tailored process, and a sample content starting point. Chapter 5 will go into the details of the newly created process add-on, followed by a validation of the work done in Chapter 6. The thesis ends in Chapter 7 by presenting a list of conclusions and further work that could be done.

The ultimate goal and contribution of this thesis is to provide software engineers with a way of incorporating the use of automated traceability into their software engineering process, in an effort to alleviate the problems surrounding traceability and maximize its potential. As a result two products are delivered. The first one is the high level process meta-model that indicates which elements need to be added to a software engineering process in order for it to support automated traceability. The second product is a sample instantiation of this meta-model, i.e. a modified process that supports automated traceability. This second product, besides exemplifying the instantiation of the meta-model, also illustrates a particular technology that can be used to model and present processes, and can be used as a starting point for organizations that wish to use its content.

This work is the continuation and completion of the early results paper “Towards a Unified Process for Automated Traceability” [6] presented on the ACM International Symposium on Grand Challenges of Traceability in Lexington Kentucky on March 2007. It forms part of the body of knowledge created in the DePaul Center for Applied Requirements Engineering lab, and it fits into a wider research initiative that aims to enhance and promote the use of automated traceability.

2. Traceability

2.1. Definition

Consider the following scenario: you are a project manager in charge of software engineering project. After a lot of effort you have compiled what you think is a complete list of requirements (at least for that moment in time). You give those requirements to your analysts and developers and they start refining the problem, coding it and testing it. The project is advancing and things are going more or less according to plan. You get called into a meeting with the primary stakeholders and they ask you the typical question: “how is the project doing?” To answer this you need to know what percentage of the requirements are fully implemented and tested. Next, they tell you that they are changing one of the requirements – a fairly common scenario. Now you need to be able to understand the impact of such a change; which code, supporting design documents, and tests will need to be modified. A discussion promptly follows, and then someone asks why a particular requirement was defined that way. Now you need to be able to identify who defined that requirement and what was the rationale behind it.

The previous scenario is common to all software engineering projects. Anyone who has worked in this field has been exposed to similar situations at one point or another.

In this scenario there is a common denominator among all the situations that arose in the meeting. All of them need for a particular characteristic to be present: **traceability**. Intuitively traceability is a way of identifying relationships between the different artifacts that are created throughout the software development lifecycle. These artifacts include work products such as requirements, use cases, classes in UML class diagrams, classes or methods, and test cases. More formally, traceability has been defined as the ability to follow the life of a requirement, in both a forwards and a backwards direction, all the way from its origin to its deployment [20]. Figure 1 illustrates this by showing a few sample traceability links that exist from the stakeholders all the way down to the unit tests.

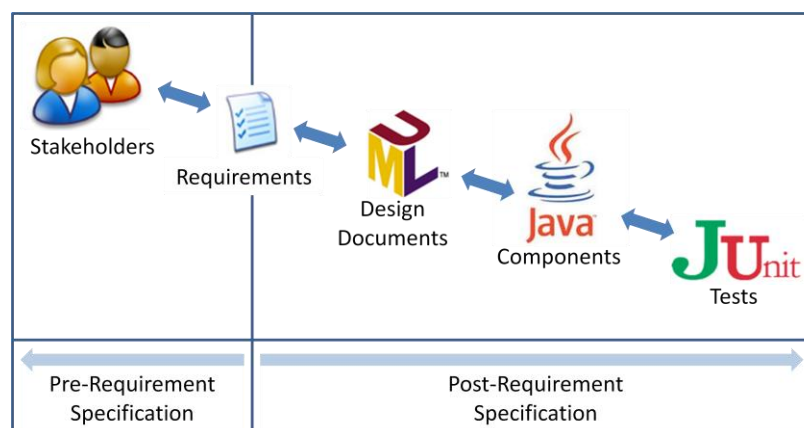


Figure 1. Sample Traceability Links



As Figure 1 illustrates, traceability is commonly divided into two phases: Pre-Requirements specification and Post-Requirements specification [20]. The reason for this division is that each phase deals with different information, and each has a different purpose and use.

The pre-requirements specification phase focuses on the issues that arise before the requirement is formally defined and written in the requirement specification [19, 20]. It deals mainly with tracing from the source of a requirement to the requirement itself and vice versa [27]. This entails tracking the rationales behind a requirement (in terms of why a requirement arose, and any assumptions and arguments that surround it) [32]; and tracking the different contributions of the stakeholders (in terms of who contributed, when, and in what capacity) [21].

The post-requirement specification phase deals with the tracing from an already specified requirement down to the artifacts that are related to it, and back up from any of those artifacts to the originating requirement [27]. The links between these artifacts and the requirements range from simple non-qualified links to much more complex and qualified relationships between artifacts [32]. For example, a simple link might be: artifact X traces to artifact Y. A complex link might look like: artifact X {depends on | is part of | evolves into | is satisfied by | is developed for | generates | is based on} artifact Y; just to name a few.

2.2. Use

After defining what traceability is, the need arises to understand how it is used. Intuitively, from the scenario presented at the beginning of this chapter, traceability provides information. This information can later be used during the software engineering tasks in a wide variety of ways.

The general consensus amongst the research community is that traceability is mainly used for the following tasks [7, 11, 15, 25, 27, 32]:

- Change impact analysis / Derivation Analysis: provides better understanding of the impact of a change to the cost, schedule and technical aspects of the project. Answers questions like: what documents, models, code modules, and tests (among other artifacts) will be impacted by a particular change?
- Coverage analysis / Compliance Verification: allows validation of which requirements have been fully implemented in the system and which ones are not. Answers questions like: which requirements have been designed, coded, tested, and deployed? This provides greater confidence in whether the objectives are being met or not, helps to understand the contribution of the work to the whole, and helps to track the progress of the project.
- Guard against gold plating: provides a mechanism to make sure that all the features that are present in a system actually correspond to a requirement; as opposed to being unnecessary features that raise the cost and risk of the project.
- Tracking rationales: if the proper information is stored, it allows an understanding of why a particular decision was made. It answers questions like: who made it, what were the alternatives, and what the pros and cons were (among other rationale information)?



- Regression testing: when a fix is introduced in the system that in turn breaks something else, traceability facilitates regression testing in order to identify what other parts of the system were affected by the change.
- Trade off analysis: when different implementation options exist, traceability facilitates a trade off analysis by allowing a comparison between the different repercussions of each option. This provides the foundation for further cost-benefit analysis.

However, not all organizations use traceability in the same way. In a seminal paper on traceability Ramesh and Jarke identified two different levels of users of traceability: low end users and high end users [32]. Low end users tend to use traceability as a mandate to comply with policies or standards, as a kind of safeguard against criticism and law suits [27]. They tend to only keep simple non-qualified links between their artifacts. On the other side of the spectrum, high end users employ a richer type system that allows them to classify and differentiate between the different types of links. They tend to view traceability as an opportunity for knowledge creation and user satisfaction. They define their trace links as products, and view them as an investment in corporate knowledge and asset management [27].

2.3. Importance

After understanding how traceability is used at a high level, it is evident the importance that it has within the software engineering activities. Traceability provides software engineers with a major source of information that they can use as a tool in their activities. But beyond the importance that comes directly from reaping the benefits of the uses listed in the previous section, there is something else to consider: traceability is required and mandated by a lot of popular software engineering, business and military standards.

As an example, the Software Engineering Institute's Capability Maturity Model Integration (CMMi) dictates that traceability is required in order to comply with the Key Process Area (KPA) of Requirements Management. This KPA is part of the staged maturity Level 2 (Managed), and more specifically it requires that an organization must [13]:

- Obtain an Understanding of Requirements
- Obtain Commitment to Requirements
- Manage Requirements Changes
- Maintain Bidirectional Traceability of Requirements
- Identify Inconsistencies between Project Work and Requirements

Note that this Key Process Area not only literally indicates that organizations must maintain bidirectional traceability links, but it also alludes in the other required practices to tasks that directly benefit from traceability. Obtaining the understanding and commitment to the requirement relies on being able to track the rationales and contribution structures. Managing the changes requires the ability to identify the impact of a change and to be able to execute regression tests. Identifying inconsistencies between



project work and requirements can only be achieved with tools that allow requirement validation and gold plating checks.

Other equally important software engineering standards such as: IEEE Standard 830-1998 – Recommended practice for software requirements specifications, and the ISO/IEC 12207 – Software Lifecycle Processes call for requirements traceability to be in place. This is a key area which this thesis addresses, since by explicitly adding traceability related tasks to a software engineering process it will be easier for organizations to comply with these standards.

There is also research that indicates that neglecting or omitting traceability has a negative impact on the overall quality of the product being developed [15]. It is understood that if there is no traceability in place more manual revisions will have to be made in order to obtain the information that traceability provides. This will directly cause an increase in cost, time, and errors. Not having it will also make the organization more prone to lose of knowledge when individuals leave, to miscommunications and misunderstandings.

2.4. Implementation

The question now arises as to how to implement this important characteristic of traceability, in order to use it effectively and reap its benefits. There are several ways to implement traceability, and each one has advantages and disadvantages over the others. In a related paper Cleland-Huang divided the different traceability implementations into several different techniques, of which the three main ones are listed below [7]:

- Simple links: Traceability is implemented via a table that illustrates the logical links between artifacts – known as a traceability matrix [37], or via other static representations such as hyper text or graphs. This is the most common method for implementing traceability, and support for this method has been implemented in several commercial tools such as Requisite Pro, Doors and Caliber. This method is simple and well understood, but very hard to set up and maintain when the number or artifacts to trace is large. Figure 2 shows a typical traceability matrix captured from the commercial tool Caliber.

	Alert when weather station fails	Easy to use	Produce De-Icing schedule	British Currency	Intuitive and Self Explanatory	Response time	New weather stations	Record readings
Alert when weather station fails	<input type="checkbox"/>							
Easy to use								
Produce De-Icing schedule								
British Currency								
Intuitive and Self Explanatory								
Response time								
New weather stations								
Record readings								

Figure 2: Sample Traceability Matrix in the Commercial Tool Caliber

- Semantically retrieved links: Traceability is implemented via tools that utilize information retrieval techniques to identify the links between artifacts based on the co-occurrence of words and terms. Chapter 3 goes into details on how this technique works.
- Executable links: Traceability is implemented via tools that define certain criteria that when met will raise an event that signifies a change to an element, which will in turn have an impact on other elements. For example this is used in simulations and models where certain non functional requirements are defined as parameters, that when changed will affect other parts of the system. This technique is also used in event based traceability [8] and in several systems that define impact analysis rules between the different elements [4, 14]. This technique is particularly good for non functional requirements.

This list is by no means exhaustive; there are other ways in which an organization could implement traceability. In fact, it has been suggested that organizations should use a combination of these techniques to make the most of their effort [7].

There are several well known and documented problems that arise when trying to implement traceability. Some of these problems are specific to one implementation technique but others apply to all of them. The following is a sample list of these common problems:

- If the links need to be identified and maintained manually it is usually very time consuming, error prone and they become outdated easily. [32]
- Usually there is no clear specification of what to trace and why. [20]
- It is difficult to document, manage and visualize the traces. Some of the more complex relationships are challenging to model. [20, 21]
- Different users have different views and ideas so it is hard for links to be defined and used consistently. [21]
- Implementing traceability can become expensive. [27]
- Sometimes it is a politically sensitive issue, where the team fears that the traces will be used against them. [27]



These problems need to be carefully considered when implementing traceability, but in spite of them the consensus is that benefits of traceability are worth it.

After this brief introduction to traceability, the following chapter will go more in depth on the particular technique of automated traceability, since this will be the base technique used throughout the paper to implement and operationalize traceability.

3. Automated Traceability

3.1. Definition

Automated Traceability is one of the possible techniques for implementing traceability. The main idea behind this technique is that it utilizes information retrieval algorithms in order to generate the traceability links automatically between the various types of software engineering work products [1, 7, 9, 11, 24, 25].

In general, the tools that implement automated traceability parse the artifacts created in the project and look for semantic similarities that could signify a dependency relationship between them. Figure 3 illustrates this by showing how traces can be identified from a requirement to several other artifacts, based on the use of similar words or phrases.

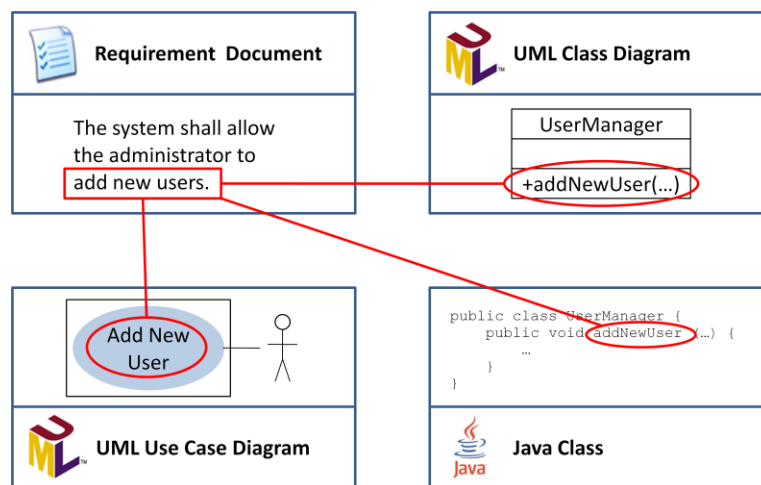


Figure 3: Schematic of Automated Traceability

More specifically, automated traceability tools make use of information retrieval models such as the *Vector Space Model* (VSM) [9, 24, 25] and the *Probabilistic Network Model* (PN) [11]. Another approach known as *Latent Semantic Indexing* (LSI) has also been used [1, 24]. In these models, traces are generated through computing a similarity score between a query (which in most cases corresponds to the text of a requirement) and each artifact in a set of traceable artifacts.

Automated traceability has been implemented in several research tools, such as Poirot¹ (developed in the Center for Requirements Engineering at DePaul University) [10] and RETRO (developed in the Department of Computer Science at the University of Kentucky) [24].

The remainder of this section illustrates the automated generation of traceability links through the use of the PN model. Prior to computing the similarity score, the words in the query and traceable artifacts

¹ Note: while most of the work of this thesis is tool independent, for the most part Poirot is used as an example.



are stemmed to their root forms and “stop” words (i.e. very common words that occur across numerous documents) are removed.

In Poirot, the PN model is implemented using the following formula to compute the basic probability of a link between a query q and a traceable artifact a as follows:

$$pr(a_j | q) = \left[\sum_{i=1}^k pr(a_j | t_i) pr(q, t_i) \right] / pr(q)$$

The first component of the formula $pr(a_j | t_i)$ is estimated as:

$$pr(a_j | t_i) = \frac{freq(a_j, t_i)}{\sum_k freq(a_j, t_k)}$$

It represents the dispersion of a term t_i within the artifact a_j , normalized over the total number of words in the artifact. The second component, $pr(q, t_i)$ is computed as:

$$pr(q, t_i) = \frac{freq(q, t_i)}{n_i}$$

Here n_i is the number of artifacts in the collection containing the term t_i . It represents the dispersion of the term t_i in the query, normalized over the total number of potential queries in which t_i occurs. The third component of the formula, $pr(q)$ is computed (using simple marginalization techniques) as:

$$pr(q) = \sum_i pr(q, t_i)$$

This represents the relevance of the term t_i to describe the query concept q ; in other words: the extent to which the term t_i describes the query concept q .

This formula belongs to the family of algorithms known as *Term Frequency – Inverse Document Frequency* (tf-idf), and it returns a probability value that is inversely proportional to the number of artifacts containing the index term, reflecting the assumption that rarer index terms are more relevant than common ones in detecting potential links. A more complete description is provided in several other papers [9, 24].

For experimental purposes, results are evaluated using the standard information retrieval metrics of recall and precision. Precision is measured as the ratio of the true links returned over the total candidate links the tool returns (signal to noise ratio); and recall is measured as the ratio of the true links returned over the total true links that exist (fraction of true relationships included) [10, 11]. These formulas are shown next:

$$precision = \frac{\text{correct links} \cap \text{true links}}{\text{retrieved links}}$$

$$recall = \frac{\text{correct links} \cap \text{true links}}{\text{correct links}}$$

In general, for most information retrieval purposes, precision is the most important metric, however for requirements traceability, recall has to be favored over precision, since industry practitioners need all true links to be identified. As there is typically a tradeoff between recall and precision, traceability tools tend to deliver high recall values at the expense of relatively low precision (i.e. many of the candidate links identified will not be true links). The alternative of favoring precision over recall is unacceptable for traceability purposes, as many true links would remain unidentified [10].



3.2. Benefits and Limitations

There are two key benefits of using automated traceability tools and techniques. The first one is the significant time savings in comparison to manually establishing a traceability matrix [10, 11]. It is not uncommon for practitioners to spend hours, days, or even weeks performing manual traceability tasks, which could be performed much more efficiently using an automated trace tool. The second key benefit is that they provide automatic support for tracing new artifacts as they are created, known as just-in-time traceability [10, 11]. Just in time traceability eliminates the risk of having to manually update a matrix each time that an artifact is added.

There is however, an important limitation to using automated traceability. Since it is based on underlying information retrieval techniques, and these are probabilistic in nature, it will never provide perfect results (100% recall with 100% precision). This limitation is one of the key motivations for this thesis, as it proposes to use this technique within a tailored software engineering process in order to improve the results. This will be explored in more detail throughout this thesis.

3.3. Experimental Results

The Center for Requirements Engineering at DePaul University has conducted several experiments with the automated traceability tool Poirot. The results for five different datasets [9, 11] are shown below in Figure 4. These results and the characteristics of each dataset are fully discussed in [11], and illustrate that in general recall of 90% is achievable at precision rates of 20-30%. A notable exception is the final dataset L&A (terse), for which the highest achievable recall was 58% at a dismal precision of 4%. The poor results achieved in this experiment were partially caused by the terseness of the data in the business and system use cases, and by the inconsistent use of a project glossary.

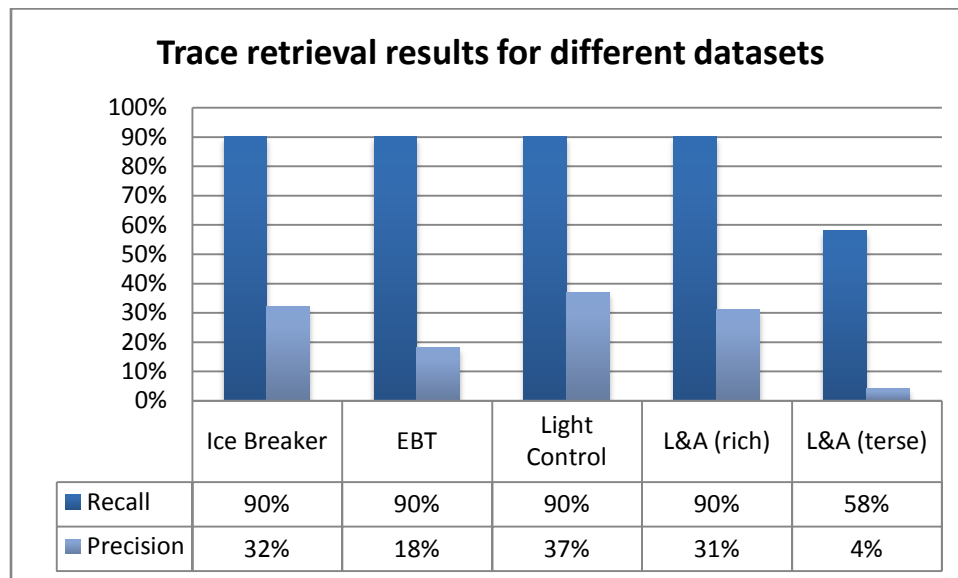


Figure 4: Experimental Results With Different Datasets Using Poirot



The fact that different data sets perform in significantly different ways when it comes to automated traceability has sparked considerable interest in the research community. Throughout these experiments the same automated traceability tools and algorithms were used, however each dataset produced different results. This can only mean that the data used in each experiment had varying degrees of quality. The positive side of this is that the quality of the artifacts that are going to be traced can be influenced by the implementation of some common best practices. The following section lists some of these best practices.

3.4. Best Practices for Automated Traceability

In a paper that addresses this issue of why different datasets perform so distinctly [9], Cleland-Huang identified a set of best practices that if implemented consistently can improve the results of the automated traceability tools. A brief explanation of these practices is listed below.

- **Trace for a purpose:** Before starting any trace implementation, identify all the artifacts and the links that will be recorded. It is important to also understand why each link is being kept and how it is going to be used. Note that this best practice is key to the success of any traceability effort, as it will provide the backbone for all traceability decisions.
- **Define a suitable trace granularity:** Again, prior to implementing traceability, decide what level of detail the trace strategy will support. For example, when tracing to code, links can be kept at the package, a class or method level. Organizations should set the level according to their information needs, bearing in mind that a finer detail may not always be beneficial.
- **Support In Place Traceability:** If the technological infrastructure permits, get the artifacts where they are created and/or stored. For example, if a CASE tool is used to track the requirements (such as Requisite Pro) set the traceability infrastructure so that it will query the requirements directly from that tool. This best practice guarantees that the latest and most up to date version of all the artifacts is kept.
- **Utilize a well defined Project Glossary:** Since automated traceability utilizes information retrieval techniques, a consistent usage of terms will improve the results of these algorithms.
- **Write quality requirements:** As the corner stone of traceability, the requirements must be of good quality. This means that they must be: correct, non ambiguous, complete, consistent, prioritized, verifiable, understandable, identifiable, etc.
- **Construct a meaningful hierarchy of information:** Keeping a good hierarchical structure between the artifacts (such as meaningful packages of classes or appropriate sub titles in the documents) can be used by the automated traceability tools to strengthen and improve its results.
- **Bridge the inter-domain semantic gap:** If within the organization the same terms are used with different meanings, the results of the automated traceability tools will not be reliable. To alleviate this, the organization should implement some kind of translation mechanism between them, prior to their use in the automated traceability tools.
- **Create rich content:** When constructing any artifact, care should be given to incorporate rationale and domain knowledge. This will create stronger links between the artifacts which in turn will improve the results. For a more in depth explanation of this best practice refer to [26]



- Utilize a process improvement plan: Automated traceability should be implemented within the greater context of a process, where it can be tested, tried and improved if necessary.

Note that most of these best practices are applicable to traceability in general, not only to automated traceability. However, since this thesis forms part of the research effort of the DePaul Center for Applied Requirements Engineering lab – a promoter of automated traceability, our main focus is to use these best practices within the context of automated traceability.

These best practices are a one of the primary foundations of this thesis, as they help organizations and project stakeholders build systems and their associated work products that are conducive to effective automated traceability.

Since actual automated traceability results have shown to be highly dependent upon the quality of the artifacts that are to be traced, these best practices need to be incorporated into the day to day work of software engineering practitioners. In other words, these best practices need to be part of the software engineering process that the organization follows.

The next two chapters will illustrate how to incorporate these best practices (among other things) into a software engineering process, as well as present a sample process add-on that was tailored specifically for automated traceability and Poirot. This is done with the goal of aiding organizations that wish to use automated traceability, so that they can maximize the benefits of this technique.



4. Process Groundwork

4.1. Benefits of a Process

Before going into the details of the created process, it is important to first explain what a process is and what the benefits of having a formal process are. This will help reinforce why we chose to create a tailored process for automated traceability.

A process, as defined in the dictionary, is a “series of actions and operations conducing to an end” [31]. Basically, a process describes the steps that are required to achieve an end result. Under this definition every individual or organization that develops software has a process, the difference lies in whether or not the process is formal. Formal processes are processes that are carefully constructed to maximize efficiency and comply with regulations or obligations; they are well documented, offer repeatable results, are supported by upper management, and are well understood by the organization. Informal processes are ad-hoc, with little or no documentation, and hence tend to be executed differently each time.

Having a formalized process in place, whether for software engineering or for any other discipline, has been highly regarded as a success factor (or a requirement) in almost all of the current business standards and methodologies; including ISO 9000, Total Quality Management, Six Sigma, COBIT, Sarbanes Oxley and CMMI. But beyond having a process just for compliance reasons, there are tangible benefits for companies that implement and follow a formal process.

One of these key benefits is that a formal process facilitates understanding and communication within the organization. By having a repository where the activities that have to be executed are detailed and explained, the personnel can be trained, a common vocabulary can be established and the overall understanding of the business will increase among the organization [28]. This in turn will help to reduce frustration among the employees and will boost their morale, improving the work environment [12].

Another benefit of having a well defined process in place is that it supports the management of the organization. It provides greater visibility into the operations, and therefore facilitates the capturing of data for measurements. Having measurements is a requirement for estimation, which improves the chances of meeting schedule deadlines and cost restrictions in future projects. Also, having a better insight into the way the organization works will help improve quality and reduce the number of defects [12].

As a tangible example of these benefits, the Software Engineering Institute (SEI) collected and combined data from several software engineering projects at Teradyne, Boeing, AIS and Hill Air Force Base [34]. In order to be able to see the effect of implementing a formal process, they gathered the data from these organizations before and after their adoption of the software engineering processes of Team Software Process (TSP) and Personal Software Process (PSP). The results are very impressive and are shown in the following set of graphs.

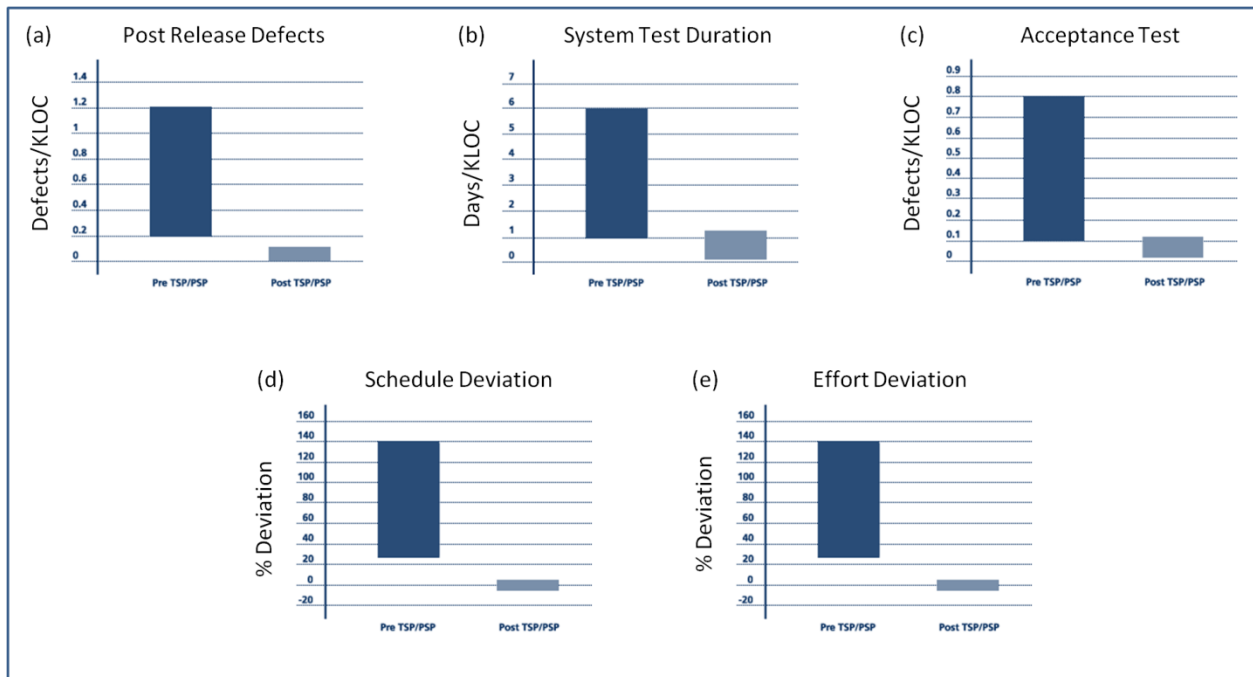


Figure 5: Benefits of Adopting a Formal Process

Graph (a) shows that, after the adoption of the TSP/PSP, the number of defects (per thousand lines of code) found after the release of the software were considerably reduced – *increasing the quality*. Graphs (b) and (c) show that the number of days spent in testing (both system and acceptance testing) decreased – *lowering the costs*. Graphs (d) and (e) show that the accuracy of the estimation of schedule and effort was greatly improved – *enabling better management*. Note that all of these graphs show a reduction both in the absolute numbers and in their ranges.

Ultimately, all of these benefits affect the bottom line, raising productivity and increasing the return on investment of the project or product [12]. Creating a software engineering process tailored for automated traceability will harness these benefits, while also improving the results that the automated traceability tools provide.

4.2. The Automated Traceability Process Meta-Model

One of the main contributions of this thesis is to provide organizations that wish to use automated traceability with a roadmap that guides them in how to incorporate this technique into their software engineering process. This roadmap takes the form of a process meta-model that will point out the key elements that organizations should add to their processes to support automated traceability.

However, in order to adapt any software engineering process for automated traceability it is important to first identify and understand the tasks related to automated traceability. In other words, the first thing needed is to determine what this process add-on will include. Note that this section starts from



the premise that the organization already has a formal software engineering process in place and that it will be augmented with the specific automated traceability tasks.

To begin with, this new process addition will need to fully support the use of traceability. Section 2.2 discussed the high level uses of traceability that have been identified in previous research [7, 11, 15, 25, 27, 32], such as: verifying or validating which requirements have been implemented, executing impact analysis when a change request comes in, identifying tradeoffs between different choices, looking up the rationale behind a decision or choice that was made, etc. These high level uses of traceability constitute the information goals that initiate a traceability analysis. Since it is outside the scope of this thesis to execute a complete usability study of traceability from the human interaction perspective, we have chosen to view traceability as an information providing service that aids the software engineering tasks. This way any task that would benefit from the information that traceability provides can use this service. The manner in which this service is used will be guided by a specific traceability strategy [9, 32].

This traceability strategy will have to be developed by the organization, and it will include what traceability links are stored, their level of granularity, and why and how they will be used. Note that this strategy is a key part of the process, since it is here that the organization asks all the important why questions: “why do we need traceability?”, “why do we want to trace to this artifact?”, “why do we see this as useful?”, etc. It is with the definition of the strategy that the organization tailors the process to its needs; hence this is what makes the process highly adaptable to different organizations, scenarios and uses of traceability.

At a lower level, this process will also need to include all the necessary activities that are required to set up and maintain the technological platform and infrastructure that will support the traceability service. Note that this is also driven by the strategy. In addition, the process needs to incorporate periodic quality control tasks, which will make sure that the automated traceability technique is providing the level of results that are expected in the organization.

And finally this tailored process will need to include guidelines and best practices that feed into the software engineering process. These guidelines will help improve the quality of the artifacts created and hence increase the probability of getting better results from the automated traceability tools.

It is from this previous list of automated traceability related tasks that we have derived the ‘Automated Traceability Process Meta-Model’. This meta-model, which points out the main elements that need to be added to a software engineering process, is shown in the following figure.

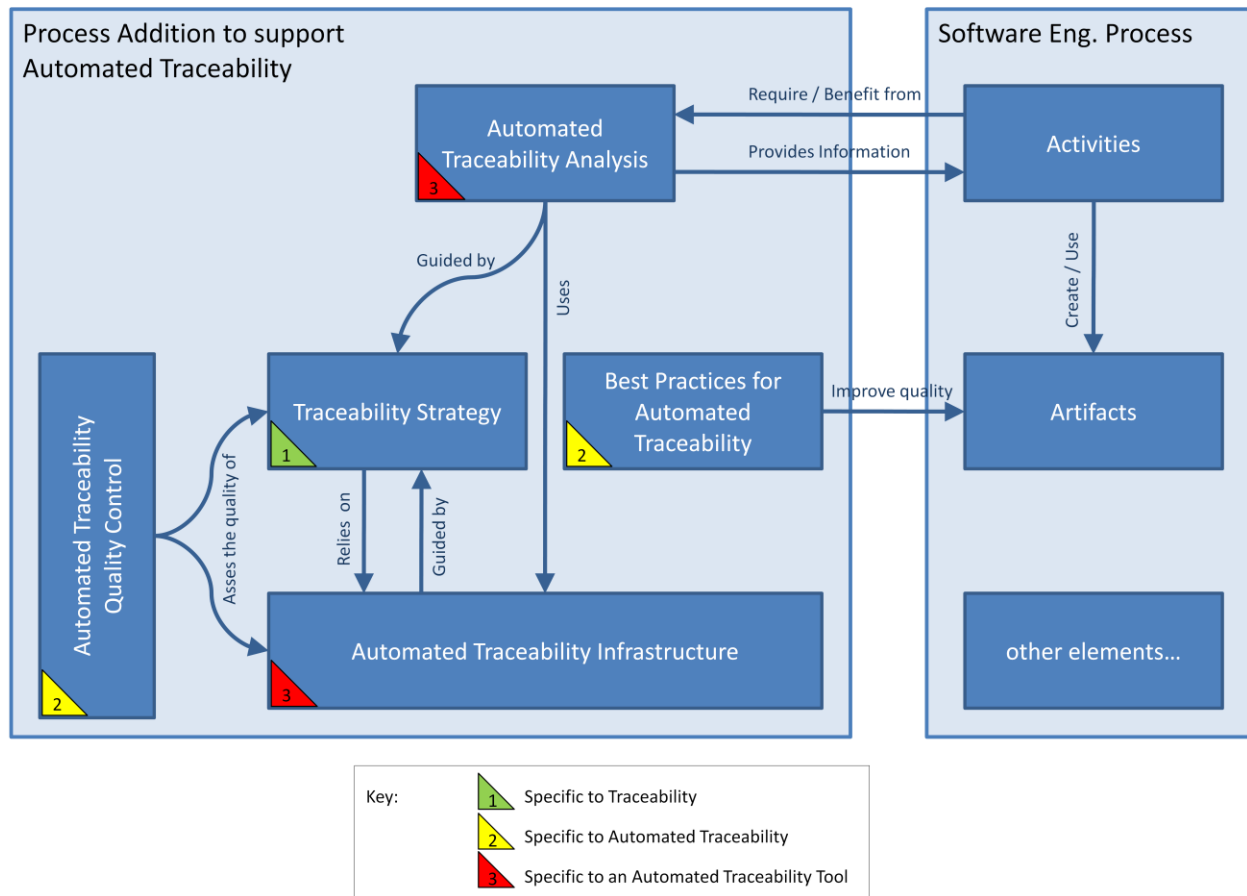


Figure 6: Automated Traceability Process Meta-Model

The idea behind this high level process meta-model is that any organization that has a formal software engineering process in place, and that wants to use automated traceability for their tracing needs, will be able to instantiate it and create an addition to their software engineering process. This meta-model is instantiated by creating the specific tasks, roles, work products and guidelines that will operationalize it, and then incorporating these into the existing organizational software engineering process.

Note that the elements in the process meta-model fall into three different categories (identified with the numbered color triangles in the bottom left corner). The first category includes items that are specific to traceability (color coded in green – number 1). The second category is comprised of elements that are specific to automated traceability (color coded in yellow – number 2). And the last category is of those elements that are specific to an automated traceability tool (color coded in red – number 3). The reason for this division is to facilitate reuse from previous instantiations of the meta-model. For example, imagine that the meta-model is instantiated and a process addition is created for the automated traceability tool Poirot. If at a later point the process engineer wishes to reuse this process to create one for RETRO, then he/she will only have to modify those elements that are specific to the tool Poirot (red elements – number 3).



The rest of this thesis will present an example of how this high level process meta-model for automated traceability was instantiated to create a specific process add-on that uses the Poirot tool as their automated traceability tool.

4.3. Paradigm for Creating Processes – Eclipse Process Framework

To instantiate the process meta-model presented in the previous section we chose to follow the process creating paradigm and use the tools provided by the Eclipse Process Framework – EPF. EPF is an open source project under the Eclipse Technology Project. It is based on the IBM Unified Method Architecture, which in turn is an evolution of the current Object Management Group (OMG) Software Process Engineering Metamodel Specification (SPEM) [18].

The main goal of EPF is to provide an extensible framework and tools for software process engineering, and to provide exemplary and extensible process content [22, 23]. Currently, in the software engineering industry there are a lot of great ideas and knowledge on how to develop software. These ideas come from different places (organizations, companies, communities, academia, research groups); and they are geared towards different technologies (.NET, J2EE), specialty domains (iterative, agile), and industries (financial, embedded, etc) [18]. The problem arises when an organization has to combine all of this knowledge and apply it to their projects. It is difficult to integrate all this information; there is redundant content, inconsistencies, isolated work, and lack of flexibility. The EPF addresses this problem by providing a standardized way of representing and managing the content and then facilitating its application within a project [18].

The content produced by the EPF is presented as a web site, which gives centralized access to the information about the practices and processes used by the organization [22, 23]. This web site allows the users to navigate and view the process through different perspectives, such as by work product, by role, and by time, among others. For example, a user can go into the web site and select a role and see the detailed description of that role, the activities in which it participates and its responsibilities. Alternatively a user can select the time perspective (lifecycle) and identify for the current stage of the process, which are the next activities, what are their inputs and outputs, and who participates in them. Note that this also serves as an educational knowledge base that can be used to train the team members.

Furthermore this web site helps to effectively execute processes in projects by bridging the gap between process management and project management [2]. When a process engineer designs a process, he/she will model the flow of the process using workflow diagrams, which the process authoring tool automatically transforms into work breakdown structures (WBS). This feature helps project managers plan and track projects based on the process. This is illustrated in the following diagram:

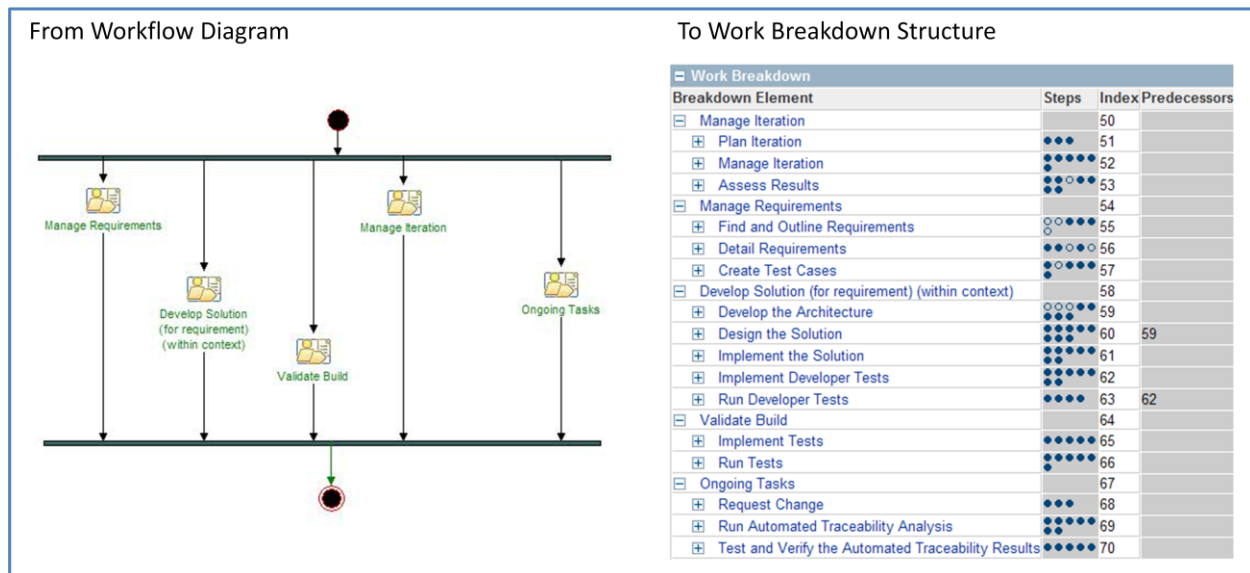


Figure 7: Bridging the Gap Between Process and Project

The EPF proposes that processes should be created in two steps. In the first step the building blocks of the process have to be defined. These building blocks are the roles (who will participate), the tasks (what needs to be done and how), the artifacts (what will be produced) and any additional guidelines. These building blocks are independent of any process, and they are called ‘Method Content’ in the EPF lingo [2, 22, 23].

Once these building blocks have been defined, the second step of the process authoring is to link and group them together in a behavioral sense. This linkage between them is what describes the lifecycle of the process, i.e. how the process will be executed through time, and it is represented as workflows and work breakdown structures. The products of this second step are what the EPF calls the ‘Process Content’ [2]. The Process Content is constructed two steps [22, 23]:

1. First ‘Capability Patterns’ are created. These represent process knowledge for a specific area, and they are composed of instances of the ‘Method Content’ optionally grouped in ‘Activities’. They can also have other process elements, such as ‘Milestones’, ‘Phases’ and ‘Iterations’.
2. Then the ‘Delivery Process’ is created. This is a grouping of instances of ‘Capability Patterns’ and it represents the complete and integrated process.

The following figure illustrates the separation of Method Content and Process Content in the EPF.

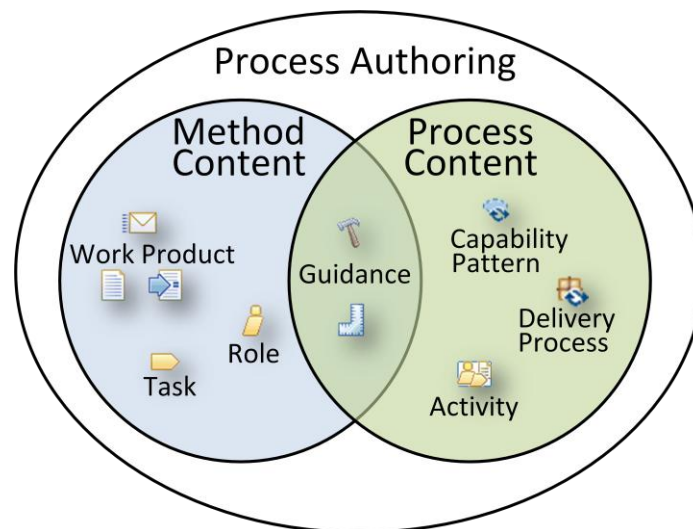


Figure 8: Separation of Method Content and Process Content [22]

The simple philosophy of process authoring proposed by the Eclipse Process Framework, along with the state of the art tool for authoring processes, the user friendly web site used to deliver the content and the other benefits that the framework and the tool provide, were the main reasons why the EPF was selected to create the sample process tailored for automated traceability and Poirot.

4.4. Content Starting Point – Basic Open Unified Process

As stated earlier, to instantiate the process meta-model presented in section 4.2 an organization needs to have a formal software engineering process in place, so that they can augment it with the specifics of automated traceability and their chosen tool. For the sample process that is being created with the EPF for Poirot we choose to use as the Basic Open Unified Process (OUP/Basic) as the starting software engineering process.

The OUP/Basic is part of the process content delivered by the Eclipse Process Framework to fulfill its goal of proving exemplary and extensible process content [22, 23]. At the time of writing the EPF is proving process content for OUP/Basic, Scrum and Extreme Programming.

In particular, the OUP/Basic is an iterative software engineering process, that claims to be minimal, complete, and extensible [2]. It is a streamlined and agile version of the Rational Unified Process, with fewer artifacts and a low level of ceremony tasks [2]. The main contributors of the OUP/Basic are important and recognized companies and institutions in the software engineering field, such as IBM, Ivar Jacobson, the European Software Institute, and the University of British Columbia among others.

The OUP/Basic is mainly intended for small teams that do not need excessive deliverables and formality. It is based on the following core principles: collaborate to align interest and understanding, balance priorities to maximize the benefits to the stakeholders, focus on architecture to mitigate risks early, and evolve continuously to get feedback and improve [2].

The content of the OUP/Basic is organized into the following four content areas, known as ‘Sub-Processes’ [2]: (note that each one corresponds to a statement in the Agile Manifesto [3] – shown in *italics*)

- Communication and Collaboration (*Individuals and interactions over processes and tools*): This is the foundation layer and it deals with the communication of the team. All of the roles are defined in this area.
- Intent (*Customer collaboration over contract negotiation*): This area deals with identifying the wants and needs of the stakeholders and making sure they are met throughout the increments of the project. Most of the tasks related to requirements and testing are defined here.
- Solution (*Working software over comprehensive documentation*): This area is about solving the problem by creating the product. It includes tasks related to analysis, design, implementation, and testing.
- Management (*Responding to change over following a plan*): This area is in charge of leading the project. It focuses on a coaching style of management where all the team members contribute and estimate their own work. It mainly includes tasks related to project management.

The work in the OUP/Basic is executed by six different roles: Stakeholder, Analyst, Tester, Developer, Architect and Project Manager. Each one of these roles has different abilities and responsibilities, and their focus aligns with the previously listed content areas [2] as shown in the following figure:

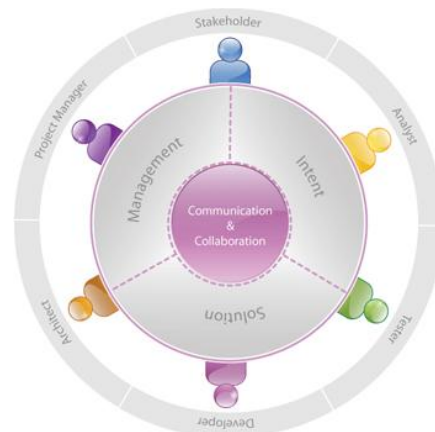


Figure 9: Content Areas and Roles of the OUP/Basic

All of these elements come together to form the Lifecycle Process which is structured into four iterations: Inception, Elaboration, Construction and Transition. This lifecycle is illustrated in the following figure:

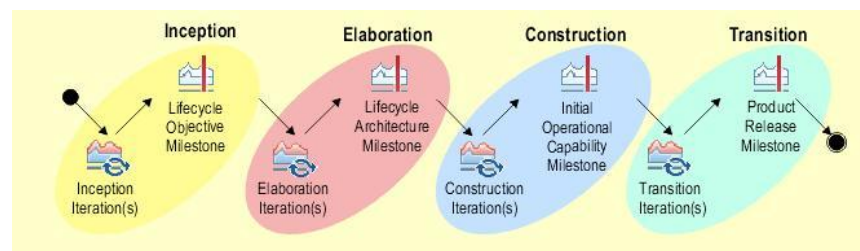


Figure 10: Lifecycle of the OUP/Basic



There are several reasons why the OUP/Basic was selected as the starting point for tailoring a process for the needs of automated traceability. The first reason is that the OUP/Basic includes a fairly complete set of initial content, including many of the proven best practices of the software engineering industry (based on the widely used and recognized Rational Unified Process – over half a million users in more than three thousand companies by 2003 [29]). This content provides a solid starting point for the additions and modifications required for automated traceability. The second reason is that, since the OUP/Basic is modeled in the EPF, its content is highly extensible and customizable. The benefit of this is twofold: it will allow for easy incorporation of the automated traceability needs into the process, as well as leave the possibility for the users of the process to further modify the rest of it to fit their needs (resizing it, integrating proprietary knowledge, etc). The third is that it benefits from the features and characteristics of the EPF such as the state of the art tool for authoring processes, and the user friendly web site used to deliver the content. And finally, the OUP/Basic is open source (under Eclipse Public License v1.0), which allows for public use and modification without problems related to copyright infringement.

Despite its virtues we are not advocating the OUP/Basic as the panacea or silver bullet in software processes. We are using the OUP/Basic solely as a content starting point, which will allow us to augment it with the specific needs of automated traceability and Poirot. Ultimately, each organization that wishes to instantiate the process meta-model is encouraged to use their institutionalized software engineering process.



5. Process for Automated Traceability

Following the paradigm of process authoring proposed by the Eclipse Process Framework, and using as a starting point the content provided by the OUP/Basic, the following subsections describe the work done to instantiate the process meta-model and create a sample process add-on for automated traceability and Poirot. This is followed by a couple of short hypothetical walkthroughs that illustrate how this sample add-on is used and how it can be modified.

Bear in mind that throughout this work the following design principles and ideas were followed:

- All of the additions and modifications had to be related to the traceability, automated traceability and Poirot. And as a whole, they had to form a cohesive set of changes.
- The work was limited to implementing what the meta-model needed, it was not meant to be a complete overhaul of the OUP/Basic.
- Any addition/modification had to be as unobtrusive possible, creating the lowest possible coupling between the OUP/Basic and the created add-on.



5.1. Building blocks – Method Content

The first part of defining the process add-on involved identifying the basic building blocks (i.e. ‘Method Content’) that would be needed. For this, a detailed revision of the ‘Method Content’ of the OUP/Basic was done. The OUP/Basic is comprised of seven roles, thirteen work products, eighteen tasks and approximately two hundred and fifty guideline elements (checklists, guidelines, examples, reports, templates, definitions, etc.). To this initial set we decided to add and modify some of them to comply with what the meta-model indicated.





5.1.1. New Method Content

The following table briefly illustrates all of the new ‘Method Content’ elements that were created for the process add-on. These additions are discussed in greater detail after the table.

Table 1: New Method Content

Type of item	Name	Scope	Brief description
Roles 	Automated Traceability Facilitator	Poirot Tool	Sets up the technical infrastructure required to support the Poirot tool.
Work Products 	Trace Strategy and Granularity	Traceability	Used to describe the different traces that the project stakeholders wish to record and the rationale supporting their decisions.
	Traceability Request	Traceability	Its purpose is to initiate a traceability query and to document the intent of the trace.



Type of item	Name	Scope	Brief description
Work Products 	Traceability Result	Poirot Tool	Represents the output generated by the traceability tool when a query is executed (in a raw or aggregated form).
Tasks 	Create Trace Strategy	Traceability	Task it that creates the Trace Strategy and Granularity work product.
	Run Automated Traceability Analysis	Poirot Tool	Details the steps required to execute an automated traceability analysis using the Poirot tool.
	Set up in-place Traceability	Poirot Tool	Task to set up the server that will run the Poirot tool, and all the required adapters that will interconnect the information repositories and case tools from which Poirot will obtain the data.
	Test and Verify the Automated Traceability Results	Automated Traceability	Provides a framework to validate the effectiveness of the results provided by the automated traceability tools.
Concepts 	Automated Traceability	Automated Traceability	Definition of Automated Traceability
	Precision	Automated Traceability	Definition of the Precision metric.
	Recall	Automated Traceability	Definition of the Recall metric.
Guidelines 	Guidelines for Creating Traceable Documents	Automated Traceability	Guideline that provides pointers to improve the artifacts that are created during the software engineering process, so that the results provided by the automated traceability tool are better.

Automated Traceability Facilitator: This role is responsible for setting up the technical infrastructure needed for Poirot. The person (or persons) who will play this role will have to install the server, install the adapters needed and provide technical support. Strong technical skills are required in Tomcat, MS SQL Server, XML and any case tool used by the organization. A screen shot of how this method content element this looks in the published web site can be viewed in Appendix 1: Automated Traceability Facilitator. Note that this role is not the one responsible for executing the trace queries. The trace queries are executed mainly by the Analyst, which is an existing role in the OUP/Basic that was enhanced to accommodate these new responsibilities (this is further explained in the next subsection).

Trace Strategy and Granularity: This work product is an artifact used to describe the different traces that the project stakeholders wish to record; it represents the organization's expectations of traceability. When the stakeholders instantiate this document they will: name all the types of links that can be identified between the different artifacts, their purpose, and the level of granularity desired [9]. For example, if the stakeholders wish to trace between a use case and a code file, they will describe this link, write down the purpose of it, and will define to which level they wish to trace to (i.e. will they trace to the code file as a whole, or will they trace down to the specific methods). The rationale behind this artifact is twofold. First, it guides the Automated Traceability Facilitator when he is setting up the technical infrastructure needed, since he will be able to easily identify the different parts that need to be interconnected. Secondly, it facilitates the process of using the results provided by the trace tool. In the case of automated traceability tools, even though they will identify the candidate links without this document, its existence will aid in determining the true links out of the candidate links. This work



product plays a key role in the process add-on, since this it defines the traceability strategy that the organization will use. A screen shot of how this method content element looks in the published web site can be viewed in Appendix 2: Trace Strategy and Granularity.

Traceability Request: This work product is used as a formality to record the intent of the trace and to initiate the workflow required to execute it. It contains information such as: who initiated the request, what information they wish to obtain from it, and why they need this information. This work product is entirely optional, and can easily be omitted. However, it is suggested to record this artifact, since it will provide useful insight into what the real traceability needs are. This information can then in turn be used by future projects to further refine the organization's Trace Strategy. A screen shot of this how method content element looks in the published web site can be viewed in Appendix 3: Traceability Request.

Traceability Results: This work product represents the results of a traceability query. It was defined as a tool specific method content element, since in its simplest form it will constitute the report that the Poirot tool creates after the candidate links have been filtered. However, any other kind of representation, such as an aggregated report created by the Analyst, will also work. These results are what is fed into the different tasks of the process that benefit/make use of traceability. A screen shot of how this method content element looks in the published web site can be viewed in Appendix 4: Traceability Results.

Create Trace Strategy: This is the task that produces the new Trace Strategy and Granularity document, it actively guides the organization in the steps required to create and document the trace strategy. The Analyst, Stakeholders and the Automated Traceability facilitator will work together to execute this task. They will start by reviewing pertinent documentation (requirements, standards, architectural diagrams, the list of work products created throughout the software engineering lifecycle, past traceability requests, etc.) and from this they will select which artifacts they wish to trace to and the different relationships between them. They will document their results in the Trace Strategy and Granularity document. Note that this is a task that can be performed on a per project basis or on a per organization basis. A screen shot of how this method content element looks in the published web site can be viewed in Appendix 5: Create Trace Strategy.

Run Automated Traceability Analysis: This is a tool specific task that describes how to use the automated traceability tool Poirot. This task includes the detailed instructions of how to run a traceability query, from logging on to the server, selecting the project, running the query, reviewing the candidate links, accepting the true links and creating the final report. This task is mainly executed by the Analyst, but any role that has the appropriate user rights can execute it. Note that this task uses the Traceability Request as its main input and produces the Traceability Results as the output. A screen shot of how this method content element looks in the published web site can be viewed in Appendix 6: Run Automated Traceability Analysis.



Set up in-place Traceability: This task, executed by the Automated Traceability Facilitator, is focused on setting up the environment to deploy the automated traceability tool Poirot. Its scope includes the installing of the servers (Tomcat, MS SQL, and Poirot) and the setting up of software adapters and local servers that enable the tool to trace into geographically distributed third party case tools. Note that this task refers the user to the System Documentation of Poirot for the low level details of how to execute the required steps. A screen shot of how this method content element looks in the published web site can be viewed in Appendix 7: Set Up In-Place Traceability.

Test and Verify the Automated Traceability Results: This is a quality control task used to measure the effectiveness of the automated traceability tool. It outlines the steps needed to conduct an experiment and evaluate the performance of the tool in terms of Precision and Recall. It starts by the Analyst selecting a manageable set of requirements and manually creating a traceability matrix. The Analyst then proceeds to trace with the tool each one of the selected requirements. After reviewing and refining the results then both matrices can be compared and Precision and Recall calculated. If the metrics are not at the organization's desired level, then a careful analysis should be performed to determine why this happened and how it can be improved. A screen shot of how this method content element looks in the published web site can be viewed in Appendix 8: Test and Verify the Automated Traceability Results.

Automated Traceability, Precision, & Recall: These method content elements are a special kind of element in the EPF called Concepts. They represent key terms and definitions that are used throughout the process. In particular, these three concepts define what automated traceability, precision and recall are. A screen shot illustrating how the concept of Automated Traceability looks in the published web site can be viewed in Appendix 9: Automated Traceability.


Guidelines for Creating Traceable Documents: This last method content element is what the EPF defines as a guideline. The purpose of this guideline is to improve the effectiveness of the automated traceability tools by making sure that all the different artifacts that are created comply with a certain level of quality. These guidelines provide the users with ideas such as: making sure that the terms in the document are used consistently with the ones defined in the glossary, that all artifacts should include rich additional content (such as rationale and domain knowledge), that everything is uniquely identifiable, and that the artifacts should be structured into meaningful hierarchies, among others [9, 26]. If followed correctly the number of shared meaningful terms that will be present in the various artifacts can increase and hence improve the results of the information retrieval techniques that the automated traceability tools use. A screen shot of how this method content element looks in the published web site can be viewed in Appendix 10: Guidelines for Creating Traceable Documents.

5.1.2. Modified Method Content



In addition to the new method content elements that were added for the process add on, a number of the existing elements of the OUP/Basic were also modified. The following table briefly lists the changes that were introduced.



Table 2: Modified Method Content

Type of item	Name	Scope	Brief description of change
Roles 	Analyst	Traceability, Automated Traceability, and Poirot Tool	Added skill related to requirements, Traceability, Automated Traceability, and Poirot tool.
	Architect	Traceability, Automated Traceability, and Poirot Tool	Added skill related to requirements, Traceability, Automated Traceability, and Poirot tool
	Project Manager	Traceability, Automated Traceability, and Poirot Tool	Added skill related to Traceability, Automated Traceability, and Poirot tool
Tasks 	Analyze Architectural Requirements		Added an optional link to the Run Automated Traceability Analysis task to aid in performing tradeoff analysis and tracking rationales.
	Assess Results		Added an optional link to the Run Automated Traceability Analysis task to aid in requirements coverage analysis.
	Create Test Cases		Added an optional link to the Run Automated Traceability Analysis task to aid in tests coverage analysis.
	Define Vision		Added an optional link to the Run Automated Traceability Analysis task to aid in identifying system constraints.
	Demonstrate the Architecture		Added an optional link to the Run Automated Traceability Analysis task to make sure that proof of concepts trace back to all the important requirements.
	Manage Iteration		Added an optional link to the Run Automated Traceability Analysis task to aid in performing impact or coverage analysis.
	Request Change		Added an optional link to the Run Automated Traceability Analysis task to aid in performing impact analysis.
Work Products 	Actor	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Architecture	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Design	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Developer Test	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Glossary	Automated Traceability	Enhanced the importance of the Glossary for automated traceability – ensuring a consistent and reliable use of vocabulary.
	Implementation	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Iteration Plan	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Project Plan	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Risk List	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
Status Assessment	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents	



Type of item	Name	Scope	Brief description of change
Work Products 	Supporting Requirements	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Test Case	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Test Log	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Test Script	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Use Case	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Use Case Model	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Vision	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
	Work Items List	Automated Traceability	Added a reference to the Guidelines for Creating Traceable Documents
Concepts 	Traceability	Traceability	Augmented the definition and provided examples of use and implementation.

The three roles that were modified – the **Analyst**, the **Architect** and the **Project Manager** – were basically augmented with traceability related skills to reflect their new responsibilities within the process add-on. Note that these changes are at the three scope levels (traceability, automated traceability and tool specific), since they include skills that range from generic requirements and traceability skills to much more specific Poirot skills.

The modifications to the tasks consist of an optional step that was added to each one. This optional step links each task with the **Run Traceability Analysis** task. The idea behind this modification is that these tasks are clear and evident examples of tasks that can benefit from the results of a traceability analysis. The optional execution of a traceability analysis will support these tasks in the executing trade-off analysis, tracking rationales, checking requirements and tests coverage, identifying constraints and performing impact analysis.

The majority of the changes to the work products consisted of adding a link in each one to the **Guidelines for Creating Traceable Documents**. This way, when the users of the process instantiate the various artifacts, they can reference the guidelines, improve their quality and ultimately increase the effectiveness of the automated traceability tools. The only different modification was the one made to the Glossary. This change consisted of reaffirming the importance of this work product in the context of automated traceability – key to ensure that the terms are used consistently and hence improve the information retrieval results.

And finally, the last modification introduced was to the concept of **Traceability**. This concept was enhanced with a more complete definition of the term, with an extended list of uses, and with a brief description of how it is usually implemented.



5.2. Linking the Building Blocks – Process Content

After defining the method content, the next step of the process authoring paradigm of EPF is to link that method content together with the rest of the process. It is in this way that the order and interaction of the tasks is explicitly declared, defining the process content. Throughout this effort close attention was paid to create an add-on that had the lowest possible coupling to the original OUP/Basic process – to promote the portability of the add-on.

For the modified method content its use within the process was left as it was. In other words, all of the modified method content is still used at the same time and place as it was defined prior to the modifications that were introduced. Our only addition to this was to explicitly indicate the optional traceability analysis step that was appended to the different tasks.

For the new method content, its integration into the OUP/Basic was more complex. The tasks of **Create Trace Strategy** and **Setup in-place Traceability** were added as steps to the activity **Initiate Project** of the **Inception Iteration**. The **Initiate Project** activity is executed at the beginning of each project, and it includes the initial planning and envisioning of the system. It is at this time that the trace strategy should be defined and the infrastructure set up. Note that these tasks do not necessarily need to be executed on a per project basis; they can be executed once and then reused in future projects.

The task of **Run Automated Traceability Analysis** was included in the **Ongoing Tasks** activity. This activity is present in every iteration of a project, and it is a placeholder for tasks that can occur at any moment during the project. This was in line with our meta-model idea of having automated traceability viewed as a service that can get called from multiple places under different circumstances. This is also the activity that includes the **Request Change** task, so it is a natural place for the **Run Automated Traceability Analysis** task. When linking this task it was characterized as an un-planned, repeatable, event driven and optional task.

The **Test and Verify the Automated Traceability Results** task was also included in the Ongoing Tasks activity. The rationale behind this decision is that this task can initiate at any time during the process, i.e. when the users feel that the automated traceability is not working as it should. However, in contrast to the **Run Automated Traceability Analysis** task, this one was characterized as a planned and optional task.

Once all the new tasks were linked into the process, their corresponding work products and executing roles were automatically integrated as well.

The following diagram illustrates how these tasks were linked into the process by showing a screenshot of the **Inception Phase**. The additions are denoted with a small arrow. Note that the **Ongoing Tasks** is present in all the other phases as well.

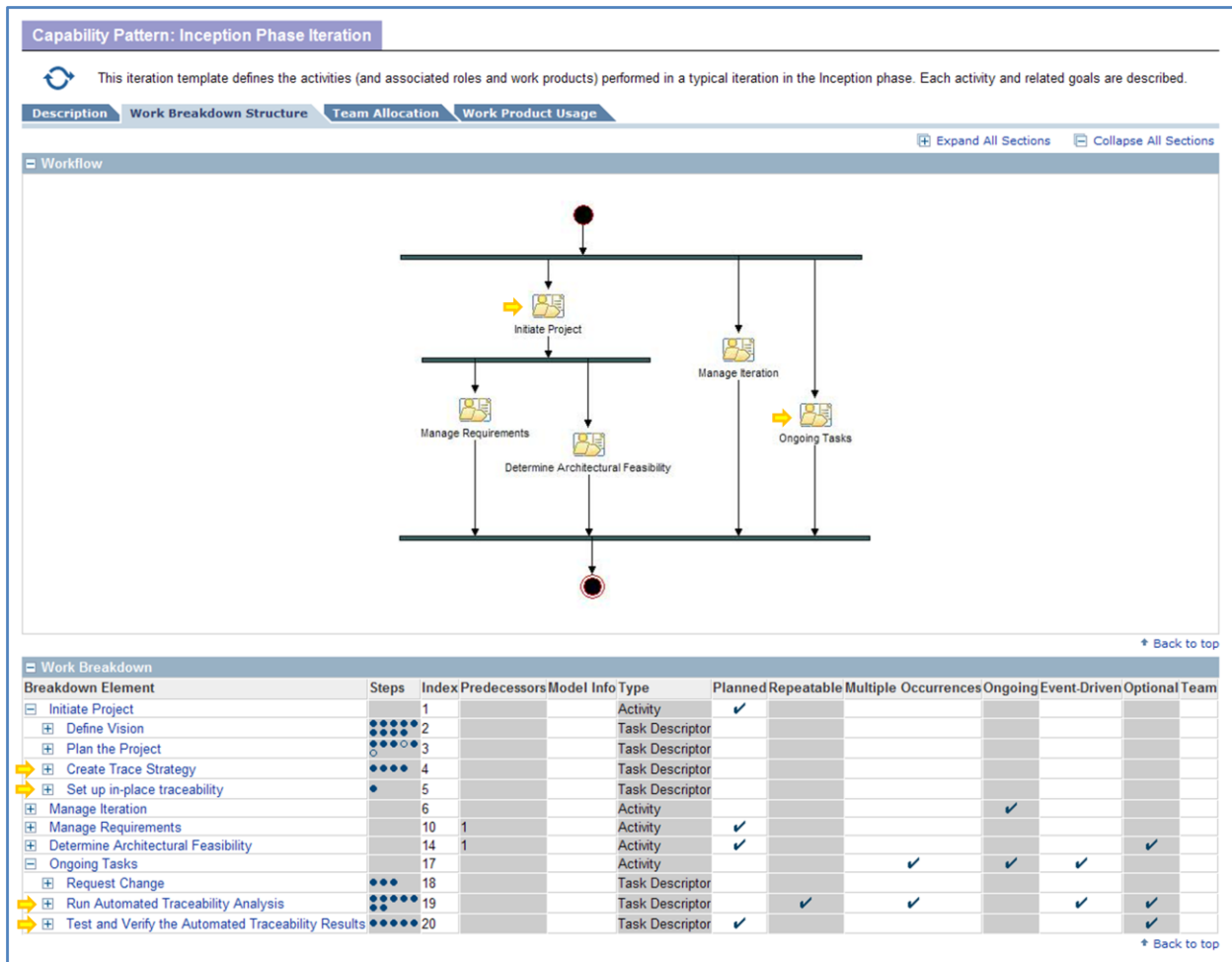


Figure 11: Linking the New Tasks Into the OUP/Basic Workflows.

5.3. Usage of the Sample Process Add-On

The question now arises as to how an organization will use this sample process add-on. To illustrate this, this subsection will present a hypothetical walk through of the software engineering process highlighting some of the elements of the process add-on, this will give the basic idea of how this is used. Note that since this process add-on was built around the OUP/Basic for Poirot, any organization that wishes to use it *as it is*, will also need to use OUP/Basic and Poirot. If the organization does not use the OUP/Basic, then they will have to instantiate the process meta-model for their own process; or if they don't want to use Poirot, then they will have to change the tool specific parts of the add-on (subsection 5.4 presents an example of this).

Now consider the scenario where a new project is about to begin, and upper management announces that, for this project, traceability will be implemented via automated traceability tools. Since the project is just starting, the team members open the process web site and in the lifecycle view they expand the **Inception** phase (illustrated in Figure 11). In the work break down structure they see that the first



activity that they have to execute is the **Initiate Project**. They drill down on this activity and discover that they need to define the vision and create an initial draft of the project plan (existing tasks of the OUP/Basic). After executing these tasks their next assignment is to execute the **Create Trace Strategy** (note that this can also be executed in parallel). For this the **Analyst**, the **Automated Traceability Facilitator**, and any relevant **Stakeholders** get together and follow the steps defined in this task in order to produce the **Trace Strategy and Granularity** work product. Upon completion of this task they ask the **Automated Traceability Facilitator** to set up the required infrastructure to support the automated traceability tool. He does this by following the steps of the **Set up in-place Traceability**.

After the initiation of the process, one of the next steps in the workflow is to **Manage the Requirements**. The first task of this activity is to **Find and Outline the Requirements**. In this task the **Analyst**, with help from the **Stakeholders**, drafts the initial requirements of the system (in the form of **Use Cases**). As he is following the guidelines and examples of how to write use cases, he sees a link to the **Guidelines for Creating Traceable Documents**. After reading these guidelines he goes back to the **Use Cases** and improves them so that they comply with the guidelines. He does this by making sure that the Use Cases include rich content, that they are uniquely identified, that they use the terms and vocabulary consistently, etc. Note that these guidelines are linked to all the work products of the process, so that when any of them is produced, its creator will be reminded to follow the guidelines.

The project progresses and the team is now in the middle of the **Construction** phase, when all of a sudden a request from management comes in indicating that they wish to change a particular requirement. The **Analyst** drills down into the **Ongoing Tasks** activity and selects the **Request Change** task. After gathering the information about the request, he sees the optional link to the **Run Automated Traceability Analysis** task, and he decides that this would be beneficial as he needs to determine the impact of the change. He follows the steps of this task and he successfully creates a report with the candidate artifacts that will likely be impacted by this change. He then returns to the **Request Change** task and proceeds to finish the task by updating the **Work Items List**. Note that in this particular example the need for a traceability query came from a change request, but in practice the need for the information that traceability provides can come from any place within the process.

This sample walkthrough was meant to illustrate at a high level how the sample process add-on is used. For the most part the rest of the additions and modifications are used in the same way.

5.4. Modification of the Sample Process Add-On

Another common question that arises is how would an organization that wishes to use another automated traceability tool use the sample process add-on? This subsection will illustrate, again via a hypothetical walkthrough, how to modify the sample process add-on for such a scenario. Note that this example also assumes that the organization is using the OUP/Basic as their software engineering process; if this is not the case then the organization would need to instantiate the meta-model for their own process.



In this hypothetical scenario an organization wishes to modify the sample process add-on in order to change the automated traceability tool from Poirot to RETRO, and to use it within an environment that employs independent validation and verification (IV&V) techniques. Before changing the sample process add-on, the first thing is to take a step back and first look at the meta-model (shown in Figure 6) from which the sample process add-on was derived.

The meta-model clearly identifies that there are two elements that are tool specific: the **Automated Traceability Analysis**, and the **Automated Traceability Infrastructure**. These two elements basically define how to use the tool and how to set it up, and clearly they will have to be changed. Also, since the scenario calls for using independent validation and verification techniques, the other element that might need to be changed is the **Automated Traceability Quality Control**. This element basically assesses the results of the automated traceability tools, and since IV&V tends to be stricter, this element will likely need to be enhanced.

The next step is to identify all of the method content elements in the sample process add-on that instantiated the previously listed meta-model elements. This is easily achieved by opening the process web site, and selecting the '**Customization for Poirot**' tab and looking through the different method content elements. In addition, each element that was added or modified has a tag that indicates to which level of scope does it belong to (specific to traceability, automated traceability or Poirot), which simplifies the search. Alternatively Table 3, which is explained in the next chapter, can also be used to identify which instantiated elements correspond to which meta-model element.

After performing this, the process engineer determines that the following elements need to change:

- Run Automated Traceability Analysis task: To list the steps required to issue queries in RETRO.
- Traceability Results document: To show how the results of a query are displayed in RETRO.
- Analyst, Architect, and Project Manager roles: To reflect any RETRO skills and responsibilities.
- Set-up In Place Traceability task: To detail how to set up RETRO's infrastructure.
- Automated Traceability Facilitator role: To reflect the skills needed to set up RETRO.
- Test and Verify Automated Traceability Results task: To add more strict controls if desired.

In addition, the process engineer might wish to create additional elements that he deems necessary – for example creating new roles that will execute the automated traceability analysis task in case they wish to restrict it, or new tasks to reflect the IV&V needs.

The next step is to execute these method content changes within the EPF process authoring tool, and to link any new elements into the process. For detailed instructions on how to do this with the EPF process authoring tool please refer to [22, 23]. Once this is finished, the new web site with the modified process can be published.

This hypothetical walkthrough illustrated the basic steps that organizations should follow if they wish to change the sample process add-on.



6. Validation of the Work

Up to this point a process meta-model has been presented that in theory will allow organizations to effectively implement automated traceability as part of their software engineering process. The instantiation of this meta-model was illustrated with a sample process add-on that was created for organizations that use the OUP/Basic as their software engineering process and that wish to use Poirot as their automated traceability tool.

This chapter will attempt to validate some of the work done, but before doing so it will relate the course by which the meta-model was derived. This will hopefully clarify some the rationale behind it. The initial attempt to create a process for automated traceability started at a low level. The initial idea was to modify the OUP/Basic. This started by reviewing the existing content of the OUP/Basic and drafting a list of the new and modified elements that needed to be added in order for this process to comply with the list of best practices presented in section 3.4 and with the needs of Poirot. When this was completed it was realized that this attempt would be of very limited use, since it required that the organizations use (or change to) the OUP/Basic and Poirot. In addition this approach would automatically inherit all of the weaknesses (and strengths) of the OUP/Basic. At this point a step back was taken, and the problem was analyzed from a process and tool independent stand point. A generic solution was needed that would allow organizations to incorporate automated traceability into any process. From this it was decided to develop a meta-model, which would serve as a roadmap/guideline for process engineers. Also, more thought was given to the other traceability related tasks that occur before and after the actual traceability analysis. These new tasks were grouped and categorized together with the method content elements that had been originally developed. These groupings became the meta-model elements. After an initial version of the meta-model was drafted, it was noticed that the elements inside of it were of different types. Some of those elements were very tool specific and others were broad enough to be useful within any traceability technique. From this it was decided that the elements should be labeled according to their scope level, which would later facilitate the reuse and modification of any process that was created from the meta-model. Finally, after all of this was done, the initial changes that were made to the OUP/Basic were revisited and completed to make sure that the sample process would include all of the elements that the meta-model indicated, hence instantiating the meta-model.

Still, after having created the meta-model and the sample process add-on, it has not yet been shown that such a meta-model is effective and complete, and that it will aid organizations that wish to use automated traceability. Nor has it been demonstrated that the sample process that was created is in fact an instance of the meta-model.

Now, it is important to mention that since processes are not an exact science and they are heavily influenced by human factors, there is no such thing as a complete scientific validation of a process. It is impossible to fully validate the effectiveness of a process. The closest thing that can be done in these cases is to have real people use the process and then have a majority of them say that the process



worked. And even then, this would still not be considered a formal proof of the process. This kind of validation would require extensive empirical research [17], which is outside the scope of this Master's thesis. This research would first need to start by clearly defining the questions that need to be answered, such as:

- Exploratory questions: Do organizations use formal processes? Do organizations use traceability? Do they need/want to change their approach to traceability? Is there interest in switching to automated traceability? What traceability tools are they using? How is the effectiveness of a process measured? How is the effectiveness of a traceability trace measured?
- Base rate questions: How often is traceability used? How often is the current traceability approach not successful? How do organizations use their formal processes? When and how is traceability used? Is there a relationship between using the process add-on and better traceability results? If there is such an improvement, is it caused solely by the process add-on?
- Design questions: How will the use of the process meta-model and the instantiated add-on improve the use of traceability in an organization?

After defining these questions, the researcher would need to identify what he/she will accept as true answers, and from there select the different research methods (controlled experiments, case studies, surveys, ethnographies, action research, etc.) and data collection techniques (brainstorming, focus groups, interviews, questionnaires, conceptual modeling, shadowing, observation, etc.) that will be used to provide the answers to the chosen questions [17]. Of course, the researcher is also going to need to have access to sufficient test subjects and test data to be able to perform the experiments.

As can be seen, a complete empirical research approach to validate the work done would require considerable experimentation, and as such it is outside the scope of this thesis – mainly due to time constraints. Instead, this chapter will use a different approach to validate the work done.

This approach will attempt to demonstrate the following:

- Show how the meta-model supports the different uses of traceability. These uses have been previously identified in research literature and are summarized in section 2.2.
- Show how the meta-model supports the best practices of automated traceability. Again these best practices are taken from previous research literature and they are listed in section 3.4.
- Show that the sample process add-on that was created for Poirot and the OUP/Basic is an instance of the meta-model.

This way, using as a foundation previous research literature, it can be shown how the meta-model (and instances of it) support traceability and improve the results of automated traceability – which are two of the key concepts of this thesis. The following subsections describe this in more detail.

6.1. Meta-Model Support for Traceability

From the list of uses of traceability presented in section 2.2 it can be seen that traceability is an information providing activity. A traceability analysis is always executed in response to a prior information need. Whether it is executed to analyze the impact of a change, or to execute a coverage analysis, or for any other reason, traceability analyses are linked to originating activities. The way in which the meta-model handles this is by following a service paradigm, in which a traceability analysis is viewed as a service that gets called upon when needed from any point within the software engineering process. This simple and fairly generic approach provides several advantages: first it promotes low coupling between the software engineering process and the add-on; and second it does not require for all the possible needs of traceability to be identified.

It is important to note that the meta-model not only supports the actual traceability analysis task, but it also supports the activities that take place prior and post the traceability analysis. Prior activities include the planning of the traceability strategy and the laying of the technical foundations needed to provide the service. Post activities include the continual improvement via the quality control tasks. Note that this is in compliance with the Plan, Do, Check, Act cycle – PDCA, which is a famous quality control and process improvement cycle popularized by Dr. Edward Deming² [30]. Deming states that any process related effort (modifications, additions, quality improvement, etc.) should iteratively follow the cycle's steps; i.e. it should start by planning the effort, executing it, checking the results, and then acting to improve them. The following figure illustrates how the meta-model tasks map to the PDCA.

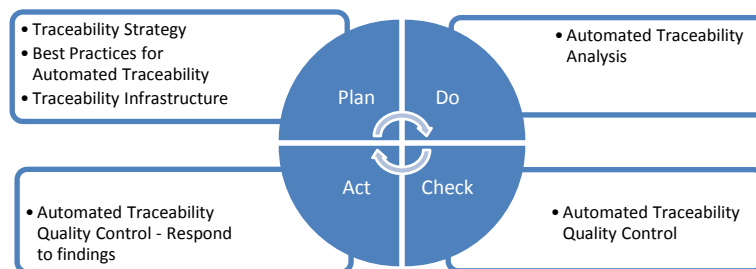


Figure 12: Mapping of Meta-Model to PDCA

By taking a service approach, in which a traceability analysis can be called upon from any point and for any reason, and by incorporating well known process improvement techniques, the meta-model fully supports the traceability needs presented in section 2.2.

6.2. Meta-Model Support of Best Practices

The main purpose of the list of best practices for automated traceability presented in section 3.4 is to improve the quality of the artifacts that are created throughout the software development process. The premise behind this is that if the quality of the artifacts is superior, then the information retrieval

² Although the PDCA cycle was popularized by Edward Deming, its creator was Walter Shewhart who was Deming's mentor at Bell Labs.























algorithms that are used by the automated traceability tools will have a better chance of identifying candidate traceability links between the created artifacts.

Given that this is a key success factor for automated traceability, the meta-model explicitly indicates that a set of best practices should be included in any process add-on that is created. However, at the meta-model level these best practices are not listed. The reason for this decision is that best practices are not a static entity; they are a continuously evolving artifact that varies depending on the context and that get improved when new information is obtained. Consequently the specific best practices have to be detailed when the meta-model is instantiated and a process add-on is created. It is in this way that the meta-model supports the best practices.

6.3. Process Add-On as Instance of Meta-Model

The last part of the validation of the work consists of demonstrating that the sample process add-on that was created for organizations that use the OUP/Basic and that wish to use Poirot as their automated traceability tool, is in fact an instance of the meta-model. This is illustrated in the following table that shows how each of the meta-model elements was instantiated in the sample process add-on:

Table 3: Mapping Between Meta-Model and Sample Process Add-On

Meta Model Element	Instantiated / Implemented as:	
Traceability Analysis		Run Automated Traceability Analysis
		Traceability Request
		Traceability Results
		All the modified tasks that link to the optional step of a Traceability Analysis t
		All the modified roles that now have additional skills and responsibilities related to traceability.
Traceability Strategy		Create Trace Strategy
		Traceability Strategy and Granularity
		The modified role of the Analyst that is in charge of creating the trace strategy.
Best Practices for Automated Traceability		Guidelines for creating traceable documents
		Automated Traceability
		All of the modified work products that link to the Guidelines
		The modified and enhanced Glossary work product
		The modified and enhanced concept of Traceability
Automated Traceability Infrastructure		Set-up in place traceability
		Automated Traceability Facilitator
		The modified role of the Project Manager that is in charge of creating the projects and users.
Automated Traceability Quality Control		Test and Verify Automated Traceability Results
		Precision
		Recall
		The modified role of the Analyst that is in charge of executing the quality control tasks.



As can be seen from the previous table, all of the elements that were created for the process add-on as well as those that were modified from the original method content of the OUP/Basic, directly map to one or more of the elements of the meta-model. It is also important to note that the best practices included in the **Guidelines for Creating Traceable Documents** include all of the ones listed in section 3.4 plus a few more; this can be seen in Appendix 10: Guidelines for Creating Traceable Documents.

This chapter demonstrated that, to the best of our knowledge, the meta-model created effectively supports the common uses of traceability (identified in research literature), and that it incorporated the current known best practices of automated traceability (also taken from previous literature). In addition, it was shown that the sample process add-on is indeed an instance of the proposed meta-model.



7. Conclusions and Further Work

Throughout this work one of the many issues that affect the requirements of a software system was explored: the issue of traceability. Traceability was defined as the ability to follow the life of a requirement, and a list was presented of how this important characteristic is used. Following this an innovative approach to implement traceability was described. This approach, called automated traceability, uses information retrieval techniques that automatically identify candidate traceability links between the different artifacts that are created in the lifecycle of a software project.

Next a set of best practices were presented that are known to improve the results of this probabilistic technique – which is highly dependent on the quality of the artifacts that are going to be traced. In order to guide potential users of automated traceability and to fully incorporate these best practices into the day to day activities of software engineers, it was suggested that automated traceability should be part of the software engineering process that the organization follows.

As such, a process add-on was created that can be “plugged” into any software engineering process that an organization uses. This process add-on, which is one of the main contributions and products of this thesis, incorporates the guidelines and best practices required to use automated traceability to its full potential. This process add-on was created in the form of a generic process meta-model that can be used under different circumstances and characteristics. Ultimately any organization that wishes to utilize this process will need to instantiate it into concrete tasks and work products.

To illustrate the instantiation of the meta-model a sample process add-on was presented. This sample process builds on the OUP/Basic software engineering process and has Poirot as its automated traceability tool. This process exemplifies the instantiation of the meta-model and it constitutes the other main product and contribution of this thesis. It also illustrates the useful process authoring paradigm proposed by the Eclipse Process Framework and it shows how processes are delivered using this framework. This sample process add-on can be viewed online at the following address: <http://castalia.cstcis.cti.depaul.edu/traceabilityprocess/>, and it can be used as an open source content starting point for organizations that wish to adopt automated traceability.

Finally, the work was validated by demonstrating how it adheres to previous research literature and how it supports the needs and best practices of traceability.

Overall, this work provides software engineers with a roadmap and sample content that they can use to incorporate automated traceability into their software engineering process – which was the main goal of this thesis.

There are several open areas of research that can be pursued from this work. The first area is that considerable empirical research can be executed to further validate the work that was done (as stated in chapter 6). This would help answer other fundamental questions about traceability that are still fairly unexplored, such as: how to enhance the experience of using traceability to fully satisfy the needs of the



user. Note that for this a multidisciplinary approach, that incorporates other fields such as the human computer interaction field, would be needed.

Also, more research needs to be done to see whether such a process meta-model is applicable both to green field projects as well as existing and legacy projects. In other words, can the plug-in be introduced to an already running project?

Another area that needs to be further explored is that of the quality control tasks that validate the effectiveness of the automated traceability tools. At this point there is a fairly good idea of how to measure the effectiveness after the fact, but there is no clear solution as to what an organization should do when the automated traceability is not providing the desired results, or how to measure it continuously during the execution of the process.

And finally, a full financial and business cost-benefit analysis should be executed. This will help understand better the pros and cons of incorporating the process add-on into existing processes. This will also improve the likelihood of this technique being adopted by industry, since it will give industry practitioners the key information and indicators that they can use to evaluate if this is something worth adopting.

As can be seen there are several areas of research that can spawn from this work. However, as it is, it represents an initial attempt to incorporate other success factors into the use of automated traceability tools. The use of automated traceability tools within the context of a process is likely to improve the returned results, which we hope will increase the appeal and adoption of this technique in industry.



8. References

- [1] Antonioli, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E. "Recovering Traceability Links between Code and Documentation". *IEEE Transactions on Software Engineering*, 28, 10 (Oct 2002), 970-983.
- [2] Balduino, R., Lyons, B. "OpenUP/Basic - A Process for Small and Agile Projects". *Eclipse Process Framework Project* (2006), http://www.eclipse.org/epf/general/OpenUP_Basic.pdf
- [3] Beck, K., et al. "Manifesto for Agile Software Development". *The Agile Alliance* (2001), <http://www.agilemanifesto.org/>
- [4] Briand, L., Labiche, Y., O'Sullivan, L. "Impact Analysis and Change Management of UML Models". *Proceedings of the International Conference on Software Maintenance* (September 2003), 256-265.
- [5] Brooks, F. "No Silver Bullet: Essence and Accidents of Software Engineering". *Computer*, 20, 4 (April 1987), 10-19.
- [6] Castro-Herrera, C., Cleland-Huang, J. "Towards a Unified Process for Automated Traceability". *ACM International Symposium on Grand Challenges of Traceability* (March 2007), 56-64.
- [7] Cleland-Huang, J., Zemot, G., Lukasik, W. "A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability". *Proceedings of the 12th IEEE International Requirements Engineering Conference* (Sep. 2004), 230-239.
- [8] Cleland-Huang, J., Chang, C., Wise, J. "Automating Performance Related Impact Analysis through Event Based Traceability". *Requirements Engineering Journal*, 8, 3 (August 2003), 171-182.
- [9] Cleland-Huang, J., Berenbach, B., Clark, S., Settimi, R., Romanova, E. "Best Practices for Automated Traceability". To appear in a future issue of *IEEE Computer*.
- [10] Cleland-Huang, J., Settimi, R., BenKhadra, O., Bezhana, E., Christina, S. "Goal Centric Traceability for Managing Non-Functional Requirements", *International Conference on Software Engineering* (2005), 363-371.
- [11] Cleland-Huang, J., Settimi, R., Chuan, D., Zou, X. "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability". *Proceedings of the 2005 International Conference on Requirements Engineering* (2005), 135-144.
- [12] CMMI Product Team. "Capability Maturity Model® Integration (CMMI®) Version 1.2 Overview". *Software Engineering Institute at Carnegie Mellon University* (2006), <http://www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview06.pdf>
- [13] CMMI Product Team. *CMMI® for Development, Version 1.2*. Carnegie Mellon University, Pittsburgh, PA, 2006.
- [14] Cysneiros, G., Zisman, A. "Tracing Agent-Oriented Systems". *ACM International Symposium on Grand Challenges of Traceability* (March 2007), 17-26.
- [15] Dömges, R., Pohl, K. "Adapting Traceability Environments to Project-Specific Needs". *Communications of the ACM*, 41, 12 (Dec. 1998), 54-62.
- [16] Dorfman, M., Thayer, R. *Software Requirements Engineering*. IEEE Computer Society Press, Los Alamitos, CA, 1997.
- [17] Easterbrook, S., Singer, J., Storey, P., Damian, D. "Empirical Research Methods in Software Engineering" **NOTE: Unpublished book**.



- [18] Eclipse Process Framework Project. "The Process Framework (EPF) Project <Beacon>". *Eclipse Process Framework Project* (2005), <http://www.eclipse.org/proposals/beamcon/>
- [19] Finkelstein, A. "Tracing Back from Requirements". *IEEE Colloquium, Computing and Control Division, Professional Group C1* (1991), 7/1-7/2.
- [20] Gotel, O., Finkelstein, A. "An Analysis of the Requirements Traceability Problem". *Proceedings First International Conference on Requirements Engineering* (1994), 94-101.
- [21] Gotel, O., Finkelstein, A. "Contribution Structures". *Proceedings of the Second International Conference on Requirements Engineering* (1995), 100-107
- [22] Haumer, P. "Eclipse Process Framework Composer – Part 1: Key Concepts". *Eclipse Process Framework Project* (2006), <http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf>
- [23] Haumer, P. "Eclipse Process Framework Composer – Part 2: Authoring method content and processes". *Eclipse Process Framework Project* (2006), <http://www.eclipse.org/epf/general/EPFComposerOverviewPart2.pdf>
- [24] Huffman Hayes, J., Dekhtyar, A., Karthikeyan, S. "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods". *IEEE Transactions on Software Engineering*, 32, 1 (Jan. 2006), 4-19.
- [25] Huffman Hayes, J., Dekhtyar, A., Osborne, J. "Improving Requirements Traceability via Information Retrieval". *Proceedings of the 11th IEEE International Requirements Engineering Conference* (Sep. 2003), 138.
- [26] Hull, E., Jackson, K., Dick, J. *Requirements Engineering*, Springer Verlag, London, UK, 2002.
- [27] Jarke, M. "Requirements Tracing". *Communications of the ACM*, 41, 12 (December 1998): 32-36.
- [28] Klingler, C. "A STARS Case Study in Process Definition". *University of Massachusetts Dartmouth* (1994), <http://www2.umassd.edu/swpi/STARS/ProcessDefStudy/trw-symposium-paper.html>
- [29] Krutchen, P. *The Rational Unified Process: An Introduction*, 3rd Edition. Addison-Wesley Professional, Reading, MA, 2003.
- [30] "PDCA". *Wikipedia* (2007), <http://en.wikipedia.org/wiki/PDCA>
- [31] "Process". *Merriam-Webster Online Dictionary* (2004), <http://www.merriam-webster.com>
- [32] Ramesh, B., Jarke, M. "Toward Reference Models for Requirements Traceability". *IEEE Transactions on Software Engineering*, 27, 1, (Jan. 2001), 58-92.
- [33] Robertson, S., Robertson, J. *Mastering the Requirements Process*. Addison-Wesley Professional, Boston, MA, 1999.
- [34] Software Engineering Institute. "Compilation Data for Projects Using TSP and PSP". *Carnegie Mellon University* (2007), <http://www.sei.cmu.edu/tsp/results/compilation.html>
- [35] The Standish Group. "Chaos Report". *The Standish Group* (1994), <http://www.standishgroup.com>
- [36] Strens, M., Sugden, R. "Change Analysis: A Step towards Meeting the Challenge of Changing Requirements". *Proceedings of the IEEE Symposium and Workshop on Engineering of Computer Based Systems* (1996), 278-283.
- [37] Weigers, K. *Software Requirements*. Microsoft Press, Redmond, WA, 2003



Appendix 1: Automated Traceability Facilitator

The screenshot shows a web browser window titled 'OpenUP / Basic + Poirot - Windows Internet Explorer'. The address bar shows the URL 'http://castalia.cstcis.cti.depaul.edu/traceabilityprocess/'. The browser's address bar includes a search engine (Google) and navigation buttons. The page content is as follows:

- Header:** 'OpenUP/BASIC + Poirot' with 'Feedback' and 'About' links.
- Left Navigation Panel:** 'Where am I | Tree Sets | OpenUP/BASIC | Customization for Poirot | Modified Elements | New Elements | Create Trace Strategy | Set up in-place traceability | Run Automated Traceability | Test and Verify the Automated Traceability | Trace Strategy and Graphical Traceability Request | Traceability Results | Automated Traceability | Precision | Recall | Guidelines for creating'.
- Main Content Area:**
 - Role: Automated Traceability Facilitator**
 - Icon: Person
 - Text: 'Sets up the technical infrastructure required to support the automated traceability tool Poirot'
 - Role Sets: Roles
 - Relationships:** A diagram showing 'Automated Traceability Facilitator' (person icon) performing 'Set up in-place traceability' (arrow icon). The relationship is labeled 'performs'.
 - Additionally Performs:**
 - Create Trace Strategy
 - Main Description:**
 - [★ New - Specific to: Poirot Tool]
 - The Automated Traceability Facilitator is in charge of setting up the technical infrastructure on which the Poirot tool will run. His main responsibilities are:
 - Installing the server
 - Installing the software adapters that will enable the tool to trace into geographically distributed third party case tools and information repositories.
 - Providing technical support
 - Staffing:**
 - Skills:** The person (or persons) who take up this role should have the following skills:
 - Strong technical experience in Tomcat and in the specific case tools that the organization uses
 - Strong working knowledge of XML files
 - Administrator rights on the Tomcat server and access to the administrators of the case tools and information repositories of the organization
 - Good customer service skills
 - More Information:**
 - Concepts:**
 - Automated Traceability



Appendix 2: Trace Strategy and Granularity

The screenshot displays a web browser window with the URL <http://castalia.ctscis.cti.depaul.edu/traceabilityprocess/>. The browser shows the 'OpenUP/Basic + Poirot' application. On the left, a navigation tree is visible with the following items: 'Where am I', 'Tree Sets', 'OpenUP/Basic', 'Customization for Poirot', 'Modified Elements', 'New Elements', 'Create Trace Strategy', 'Set up in-place traceability', 'Run Automated Traceability', 'Test and Verify the Automated Traceability Results', 'Trace Strategy and Granularity', 'Traceability Request', 'Traceability Results', 'Automated Traceability', 'Precision', 'Recall', and 'Guidelines for creating'. The main content area is titled 'Artifact: Trace Strategy and Granularity' and contains the following sections:

- Domain:** Requirements
- Purpose:** [★ New - Specific to: Traceability] This work product is used to describe the different traces that the project stakeholders wish to record (and the rationale supporting their decisions). It supports the idea that the organization should understand the purpose and reason for tracing.
- Relationships:**

Roles	Responsible:	Modified By: <ul style="list-style-type: none">Analyst
Tasks	Input To: <ul style="list-style-type: none">Test and Verify the Automated Traceability ResultsRun Automated Traceability Analysis	Output From: <ul style="list-style-type: none">Create Trace Strategy
- Description:**

Brief Outline	A possible outline for this document is: <ul style="list-style-type: none">Traceable artifacts:<ul style="list-style-type: none">NameLocationLevel of granularityAdaptersTraceability links:<ul style="list-style-type: none">NameRationale
Main Description	When the stakeholders instantiate this document they will: name all the artifacts that they want to trace to, identify their location, list the types of links that can be identified between them, the purpose of each link, and the level of granularity desired in each trace.



Continuation of Trace Strategy and Granularity:

Impact of not having Even though certain traceability techniques like automated traceability support after-the-fact traceability, having this document is highly recommended. This since it will provide guidelines that will help the Requirements Analyst determine which returned links are relevant, reducing the time required to sort through them.

Representation Options This artifact should be represented in a document form, that should be accessible and known by the organization.
It is highly recommended that it should include a traceability diagram that illustrates the whole traceability scheme. An example of such diagram could be:

Legend:
B: Business motivated links
IT: IT motivated Links
O: Organizational wide-motivated Links

WORD DOCUMENTS

- Business Goal (BG)
- Stakeholder Request (SR)
- Business Use Case Life Cycle Stage (BUCLS)

REQUIREMENTS DATABASE

- Minimal Marketable Feature (MMF)
- MMF Group (MMF_Grp)
- Business Use Case (BUC)

DESIGN MODELING TOOL

- System Use Case (SUC)

DOCUMENT VERSIONING SYSTEM

- Product

Relationships:

- Business Goal (BG) $\xrightarrow{1..*}$ Stakeholder Request (SR) (B) Which SRs address which BG? Must be Traced to
- Stakeholder Request (SR) $\xrightarrow{1..*}$ Minimal Marketable Feature (MMF) (B) Which MMFs satisfy which SR(s)? Will be realized by Existing or New
- Minimal Marketable Feature (MMF) $\xrightarrow{2..*}$ MMF Group (MMF_Grp) (B) Which MMFs are members of one (or more) logical group of features?
- Stakeholder Request (SR) $\xrightarrow{1..*}$ Business Use Case (BUC) (B) Which SRs are needed within which BUC(s)?
- Business Use Case Life Cycle Stage (BUCLS) $\xrightarrow{1..*}$ Business Use Case (BUC) (B) Which BUCs occur within which BUCLSs?
- Business Use Case (BUC) $\xrightarrow{1..*}$ System Use Case (SUC) (O) Which BUCs require which SUCs? Must be Traced to
- System Use Case (SUC) $\xrightarrow{1..*}$ Product (O) Which Products and Releases contain which CSCs (and CSComps)? Is Realized by
- System Use Case (SUC) $\xrightarrow{1..*}$ Stakeholder Request (SR) (IT) Which SUCs implement which CSCs (and CSComps)?
- Business Use Case (BUC) $\xrightarrow{1..*}$ Stakeholder Request (SR) (B) Which SRs are needed within which BUC(s)?
- Business Use Case (BUC) $\xrightarrow{1..*}$ Business Use Case Life Cycle Stage (BUCLS) (B) Which BUCs occur within which BUCLSs?
- System Use Case (SUC) $\xrightarrow{1..*}$ Business Use Case (BUC) Hierarchical
- Product $\xrightarrow{1..*}$ System Use Case (SUC) Hierarchical



Appendix 3: Traceability Request

The screenshot displays a web browser window titled 'OpenUP / Basic + Poirot - Windows Internet Explorer'. The address bar shows the URL 'http://castalia.ctcis.cti.depaul.edu/traceabilityprocess/'. The browser's toolbar includes a search box with 'Google' and various navigation icons. The page header features the 'OpenUP/Basic + Poirot' logo and navigation links for 'Feedback' and 'About'. A 'Print' button is also visible.

The main content area is titled 'Artifact: Traceability Request'. It contains the following information:

- Request that originates an traceability query.**
Domain: Change Management
- Purpose**
[* New - Specific to: Traceability]
The purpose of the Trace Request is to initiate an traceability query.
It also serves to document the intent of the trace. This will help future projects in determining the real needs of traceability and create Trace Strategy and Granularity documents that will suffice the information needs of the organization.
- Relationships**

Roles	Responsible:	Modified By: <ul style="list-style-type: none">AnalystAny RoleArchitectProject ManagerTester
Tasks	Input To: <ul style="list-style-type: none">Run Automated Traceability Analysis	Output From: <ul style="list-style-type: none">Analyze Architectural RequirementsAssess ResultsCreate Test CasesDefine VisionDemonstrate the ArchitectureManage IterationRequest Change
- Description**

Brief Outline	This document includes information such as: <ul style="list-style-type: none">Who initiated the request?What information do they require?Why do they need that information?
----------------------	---

The browser's status bar at the bottom shows 'Internet' and a zoom level of '100%'.



Appendix 4: Traceability Results

The screenshot shows a web browser window titled 'OpenUP / Basic + Poirot - Windows Internet Explorer'. The address bar shows the URL 'http://castalia.ctscis.cti.depaul.edu/traceabilityprocess/'. The browser's address bar includes a search engine (Google) and navigation buttons. The application header is 'OpenUP/Basic + Poirot' with 'Feedback' and 'About' links. A 'McAfee SiteAdvisor' icon is visible in the top left. The left sidebar contains a 'Where am I' section with a 'Tree Sets' icon, and a 'Customization for Poirot' section with a tree view of elements including 'Modified Elements', 'New Elements', 'Create Trace Strategy', 'Set up in-place traceab', 'Run Automated Tracea', 'Test and Verify the Auto', 'Trace Strategy and Gra', 'Traceability Request', 'Traceability Results', 'Automated Traceability', 'Automated Traceability', 'Precision', 'Recall', and 'Guidelines for creating'. The main content area is titled 'Artifact: Traceability Results' and contains the following sections:

- Artifact: Traceability Results**
 - The results of an traceability query or queries
 - Domain: Change Management
 - Expand All Sections / Collapse All Sections
- Relationships**

Roles	Responsible:	Modified By: <ul style="list-style-type: none">Analyst
Tasks	Input To: <ul style="list-style-type: none">Analyze Architectural RequirementsAssess ResultsCreate Test CasesDefine VisionDemonstrate the ArchitectureManage IterationRequest Change	Output From: <ul style="list-style-type: none">Run Automated Traceability Analysis
- Description**
 - Main Description**: [★ New - Specific to: Traceability, Poirot]
 - The Traceability Results document is the output generated by the traceability tool when a query is executed. In the case of using Poirot, It can be the simple report that is generated when clicking on 'Report' after filtering the accepted links, or it can be an ad-hoc aggregated report produced by an Analyst after executing one or more queries.
- Tailoring**
 - Representation Options**: If the default report that Poirot outputs is used, it will look like:

Accepted Links
(Report Generated : 2007/0/4)

PROJECT : IBS

Trace artifact # 9014 " Data received from weather stations shall be updated regularly "
Against: Class Diagram.

Row	Document	Document	Confidence Level
NO: ID:	Description:		



Appendix 5: Create Trace Strategy

The screenshot displays a web browser window titled 'OpenUP / Basic + Poirot - Windows Internet Explorer'. The address bar shows the URL 'http://castalia.ctcis.cti.depaul.edu/traceabilityprocess/'. The browser's toolbar includes a search box with 'Google' and various navigation icons. The page header features the 'OpenUP/Basic + Poirot' logo and navigation links for 'Feedback' and 'About'. A left-hand navigation pane shows a tree structure with 'Create Trace Strategy' selected. The main content area is titled 'Task: Create Trace Strategy' and contains the following sections:

- Task: Create Trace Strategy**
 - This task lists the steps necessary to create the Trace Strategy and Granularity document.
 - Disciplines: Requirements
 - Expand All Sections / Collapse All Sections
- Purpose**
 - [* New - Specific to: Traceability]
 - The purpose of this task it to create the Trace Strategy and Granularity document, which is a very important document that defines the different traces that are of interest to the stakeholders and the rationale behind each kind of trace.
 - Back to top
- Relationships**

Roles	Primary Performer: <ul style="list-style-type: none">Analyst	Additional Performers: <ul style="list-style-type: none">Automated Traceability FacilitatorStakeholder
Outputs	<ul style="list-style-type: none">Trace Strategy and Granularity	

- Back to top

- Main Description**
- During this activity the Analyst, the Stakeholders and the Automated Traceability Facilitator will brainstorm and define the Trace Strategy and Granularity document.
- They will review the work products of the software engineering process, the corporate standards and regulations, the architecture of their information repositories and case tools; and with this information they will identify and document which links are needed to fully support their tracing needs.
- Back to top
- Steps**
- Expand All Steps / Collapse All Steps
- Review documentation**
 - Start by reviewing all the applicable documentation, such as:
 - Any specific requirements for traceability for this project
 - Any corporate standard, mandate or certification requirement for traceability
 - The architecture diagrams of the information repositories and case tools
 - The list of all the work products that are created by the software engineering process
 - Traceability Requests from past projects



Continuation of Create Trace Strategy:

The screenshot shows a web browser window titled "OpenUP / Basic + Poirot - Windows Internet Explorer". The address bar shows the URL "http://castalia.ctcis.cti.depaul.edu/traceabilityprocess/". The page content is organized into several sections:

- Where am I / Tree Sets:** A navigation tree on the left side of the page.
- OpenUP/Basic + Poirot:** The main header of the application.
- Customization for Poirot:** A section containing a list of "New Elements" such as "Create Trace Strategy", "Set up in-place traceability", "Run Automated Traceability", "Test and Verify the Automated Traceability", "Trace Strategy and Guidelines", "Traceability Request", "Traceability Results", "Automated Traceability", "Automated Traceability Precision", "Recall", and "Guidelines for creating traceable documents".
- Select artifacts to trace:** A section with instructions: "From the list of work product and taking into consideration the supporting documentation, select the artifacts that your organization wants to trace. For each artifact document the following:" followed by a list: "Its name", "Its physical location", "The level of granularity that you want to trace to. For example: if the artifact is code, the level of granularity can be at the Package, Class, Method, or even Line level of granularity", and "If there are existing adapters for this artifact at the desired granularity, or if new ones need to be created."
- Define trace relationships:** A section with instructions: "From the selected artifacts identify the links between them that you represent the traces your organization is interested in. Note that a careful balance between benefit and cost should be achieved, since there is little benefit in trying to trace from every artifact to everything else. For each trace link identified make sure to document the following:" followed by a list: "Name of the link" and "Purpose of the link".
- Create traceability diagram:** A section with instructions: "Once all the information has been collected, try to create a diagram that shows all relevant artifacts and the traceability links between them. This diagram will facilitate the setup of the technical infrastructure required as well as the daily usage of the traceability tools."
- More Information:** A table with two rows: "Concepts" (Automated Traceability, Traceability) and "Guidelines" (Guidelines for creating traceable documents).



Appendix 6: Run Automated Traceability Analysis

Task: Run Automated Traceability Analysis

This task details the steps required to execute an automated traceability analysis using the Poirot tool.
Disciplines: Configuration & Change Management

Purpose
[★ New - Specific to: Poirot Tool]
The purpose of this task is to identify all the possible artifacts that trace down from a specific requirement or query.

Relationships		
Roles	Primary Performer: <ul style="list-style-type: none">Analyst	Additional Performers: <ul style="list-style-type: none">Any Role
Inputs	Mandatory: <ul style="list-style-type: none">Traceability Request	Optional: <ul style="list-style-type: none">Trace Strategy and Granularity
Outputs	<ul style="list-style-type: none">Traceability Results	

Main Description
Given a request for a traceability analysis, you will be able to follow the steps provided in this task to run it using the Poirot tool. Poirot will provide you with a list of possible artifacts that satisfy your query, and you will then proceed to examine them and confirm the true links. Once you are satisfied with the results, the task will indicate how to output the results.

Steps

- Access the Poirot web site**
Access your organization's Poirot server via an Internet browser (such as Internet Explorer or Firefox) by typing in the appropriate URL. If you don't know the URL please contact your Automated Traceability Facilitator so that he/she can tell it to you.
- Log in**
Once you have accessed the Poirot server, by default the following log in page will come up:



Continuation of Run Automated Traceability Analysis

OpenUP / Basic + Poirot - Windows Internet Explorer

http://castalia.cstcis.cti.depaul.edu/traceabilityprocess/

OpenUP/Basic + Poirot

Where am I | Tree Sets

OpenUP/Basic

Customization for Poirot

Modified Elements

New Elements

Create Trace Strategy

Set up in-place traceab

Run Automated Tracea

Test and Verify the Auto

Trace Strategy and Gra

Traceability Request

Traceability Results

Automated Traceability

Automated Traceability

Precision

Recall

Guidelines for creating

Run the query

Next the query screen comes up:

Poirot : TraceMaker

Project Query Artifacts Options Help

IBS > Query

Query for Artifact Document:

Enter artifact ID# or pick from the list

Use Freeform

Against Artifact Type:

Class Diagram
Sequence Diagram
Requirement
NFR
Business Requirements

add>
add all>>

<remove
<<remove all

Run Query

To run a query you first need to either: enter an artifact id; or select an artifact type and choose a particular artifact from the new dropdown that will come up; or type in a free form text will all the words that you wish to trace from.

Then you must select the type of artifact that you are tracing to. Here you can optionally select all of them for a complete trace. Please review the Trace Strategy and Granularity document to help you identify which traces are useful to you.

Finally you must click on the 'Run Query' button to execute it.

Review returned links

After clicking on the 'Run Query' a screen with returned links will be shown similar to:



Appendix 7: Set Up In-Place Traceability

The screenshot shows a web browser window with the address bar displaying `http://castalia.ctscis.cti.depaul.edu/traceabilityprocess/`. The browser title is "OpenUP / Basic + Poirot - Windows Internet Explorer". The page content is as follows:

- Task: Set up in-place traceability**
 - This tasks lists the steps required to set up the infrastructure for in-place automated traceability.
 - Disciplines: Requirements
 - Expand All Sections / Collapse All Sections
- Purpose**
 - [★ New - Specific to: Poirot Tool]
 - The purpose of this task is to set up the server that will run the Poirot tool, and all the required adapters that will interconnect the information repositories and case tools from which Poirot will obtain the data.
 - Back to top
- Relationships**

Roles	Primary Performer: <ul style="list-style-type: none">Automated Traceability Facilitator	Additional Performers:
--------------	---	------------------------

Back to top
- Main Description**

During this activity the Automated Traceability Facilitator will install the Poirot sever, the Tomcat server and the SQL Server where Poirot will run. In addition , for each artifact type that will be traced, an adapter and a local sever component will also be installed. This will be done in accordance to what the Trace Strategy and Granularity document indicates that is needed.

Back to top
- Steps**
 - Expand All Steps / Collapse All Steps
 - Refer to System Documentation**

For a detailed description of the steps required to execute this task, please refer to the Poirot System - Developers Guide.
 - Back to top
- More Information**

Concepts	<ul style="list-style-type: none">Automated TraceabilityTraceability
-----------------	---

Back to top

The status bar at the bottom shows the URL `http://castalia.ctscis.cti.depaul.edu/traceabilityprocess/poirot_plugin/tasks/set_up_in_place_traceability_qf7uoNSREduxnIjy8DN` and the page is viewed at 100% zoom.



Appendix 8: Test and Verify the Automated Traceability Results

The screenshot shows a web browser window with the URL <http://castalia.ctcis.cti.depaul.edu/traceabilityprocess/>. The page title is "OpenUP/Basic + Poirot". The main content area is titled "Task: Test and Verify the Automated Traceability Results".

Task: Test and Verify the Automated Traceability Results

This task provides some guidelines to test and verify the effectiveness of the results provided by the automated traceability tool.

Disciplines: Requirements

Purpose

[* New - Specific to: Automated Traceability]

The purpose of this task is to provide a framework to validate the effectiveness of the results provided by Poirot.

Relationships

Roles	Primary Performer:	Additional Performers:
	<ul style="list-style-type: none">Analyst	
Inputs	Mandatory:	Optional:
	<ul style="list-style-type: none">Trace Strategy and Granularity	<ul style="list-style-type: none">None

Main Description

The effectiveness of automated traceability tools is measured in terms of Precision and Recall. There is an inherent trade-off between these metrics, and for traceability purposes recall has to be favored over precision.

During this task, the effectiveness of the tool will be measured. This is a task that has to be executed periodically to make sure that it is performing according to the needs of the organization.

Steps

Obtain a Result Set

First select a manageable set of requirements and manually create their traceability matrix. Be as thorough and detailed as possible, since this will constitute the 'yard stick' against which you will measure the effectiveness of the automated traceability tool. Make an effort to include all the artifacts that relate to those requirements.

Run Automated Traceability queries

For each of the requirements selected, run an automated traceability query. Review all the results, and provide the feedback – assuring the tool which results correspond to true links. Record all the links returned in a separate traceability matrix.



Continuation of Test and Verify the Automated Traceability Results

OpenUP / Basic + Poirot - Windows Internet Explorer

http://castalia.ctcis.cti.depaul.edu/traceabilityprocess/

McAfee SiteAdvisor

OpenUP/Basic + Poirot

OpenUP/Basic + Poirot

Feedback | About

Print

Where am I | Tree Sets

OpenUP/Basic

Customization for Poirot

- Modified Elements
- New Elements
 - Create Trace Strategy
 - Set up in-place traceab
 - Run Automated Tracea
 - Test and Verify the Auto**
 - Trace Strategy and Gra
 - Traceability Request
 - Traceability Results
 - Automated Traceability
 - Automated Traceability
 - Precision
 - Recall
 - Guidelines for creating

During this task, the effectiveness of the tool will be measured. This is a task that has to be executed periodically to make sure that it is performing according to the needs of the organization.

Back to top

Steps

Expand All Steps | Collapse All Steps

Obtain a Result Set

First select a manageable set of requirements and manually create their traceability matrix. Be as thorough and detailed as possible, since this will constitute the 'yard stick' against which you will measure the effectiveness of the automated traceability tool. Make an effort to include all the artifacts that relate to those requirements.

Run Automated Traceability queries

For each of the requirements selected, run an automated traceability query. Review all the results, and provide the feedback – assuring the tool which results correspond to true links. Record all the links returned in a separate traceability matrix.

Evaluate results

Compare the two traceability matrices – the one created manually and the one with the results from the automated traceability tool. Review all the mismatches. Paying careful attention to those links returned by the automated traceability tool that were not identified manually; these could be false positives but may also indicate mistakes made by the analyst. If necessary correct and complete the manual traceability matrix.

Calculate metrics

Calculate the metrics for precision and recall using the formulas provided in the guidelines.

Identify improvement opportunities

If the metrics for precision and recall don't satisfy your organizational goals, then a detailed evaluation should be performed to investigate why some links were not returned. Refer to the guidelines for Traceable Documents to identify some possible reasons. At this point an action and improvement plan should be created.

Back to top

More Information

Concepts	<ul style="list-style-type: none">Automated TraceabilityPrecisionRecallTraceability
Guidelines	<ul style="list-style-type: none">Guidelines for creating traceable documents

Back to top

Internet 100%



Appendix 9: Automated Traceability

The screenshot shows a web browser window with the URL <http://castalia.ctcis.cti.depaul.edu/traceabilityprocess/>. The browser title is "OpenUP / Basic + Poirot - Windows Internet Explorer". The page content is as follows:

OpenUP/Basic + Poirot

Where am I | Tree Sets

- OpenUP/Basic
 - Customization for Poirot
 - Modified Elements
 - New Elements
 - Create Trace Strategy
 - Set up in-place traceab
 - Run Automated Tracea
 - Test and Verify the Auto
 - Trace Strategy and Gra
 - Traceability Request
 - Traceability Results
 - Automated Traceability
 - Automated Traceability
 - Precision
 - Recall
 - Guidelines for creating

Concept: Automated Traceability

Expand All Sections | Collapse All Sections

Relationships

Related Elements	Related Elements
	<ul style="list-style-type: none">PrecisionRecallArchitectureDesignDeveloper TestImplementationIteration PlanProject PlanRisk ListStatus AssessmentWork Items ListActorGlossarySupporting RequirementsUse CaseUse Case ModelVisionTest CaseTest LogTest ScriptDefine VisionTraceabilityGuidelines for creating traceable documentsAutomated Traceability FacilitatorSet up in-place traceabilityTraceability ResultsCreate Trace StrategyRun Automated Traceability AnalysisTest and Verify the Automated Traceability ResultsTrace Strategy and GranularityTraceability Request

Main Description

[★ New - Specific to: Automated Traceability]

Automated traceability methods commonly utilize information retrieval techniques to generate traceability links between various types of software engineering work products. These work products include artifacts such as requirements, use cases, classes in UML class diagrams, classes or methods, and test cases.

In general, the tools that implement automated traceability parse the artifacts created in the project and look for semantic similarities that could signify a dependency relationship between

Continuation of Automated Traceability

The screenshot displays a web browser window with the URL `http://castalia.cstcis.cti.depaul.edu/traceabilityprocess/`. The browser shows a page titled "OpenUP/Basic + Poirot" with a navigation menu on the left. The main content area is titled "Main Description" and contains the following text:

[* New - Specific to: Automated Traceability]

Automated traceability methods commonly utilize information retrieval techniques to generate traceability links between various types of software engineering work products. These work products include artifacts such as requirements, use cases, classes in UML class diagrams, classes or methods, and test cases.

In general, the tools that implement automated traceability parse the artifacts created in the project and look for semantic similarities that could signify a dependency relationship between them. The following figure illustrates this by showing how traces can be identified from a requirement to several other artifacts, based on the use of similar words or phrases:

The diagram shows a requirement box on the left: "Requirement 01: The system shall allow the admin to add new users". A red line connects the phrase "add new users" in the requirement to the "addNewUser(...)" method in a "userManager" class box on the right. Below the requirement box is a separate box containing "Add New User" in a blue oval, with a red line connecting it to the "addNewUser(...)" method. A stick figure is also present in the diagram.

More specifically, automated traceability tools make use of information retrieval models such as the Vector Space Model (VSM) and the Probabilistic Network model (PN). Another approach known as Latent Semantic Indexing (LSI) has also been used. Traces are generated through computing a similarity score between a query (which in most cases corresponds to the text of a requirement) and each artifact in a set of traceable artifacts.

More Information

Concepts

- Precision
- Recall
- Traceability



Appendix 10: Guidelines for Creating Traceable Documents

The screenshot shows a web browser window titled 'OpenUP / Basic + Poirot - Windows Internet Explorer'. The address bar shows the URL 'http://castalia.cstcis.cti.depaul.edu/traceabilityprocess/'. The page title is 'OpenUP/Basic + Poirot'. The main content area is titled 'Guideline: Guidelines for creating traceable documents'. It contains a paragraph explaining the importance of artifact quality for automated traceability tools. Below this is a 'Relationships' section with a 'Related Elements' list. The 'Main Description' section provides specific instructions for creating artifacts.

Guideline: Guidelines for creating traceable documents

In order to maximize the effectiveness of the automated traceability tools, the artifacts to be traced should meet a certain level of quality. This is because automated traceability uses information retrieval techniques, and these rely heavily on the text of the artifacts. This guideline will provide pointers to improve the artifacts that are created during the software engineering process, so that the results provided by the automated traceability tool are better.

Relationships

Related Elements

- Create Trace Strategy
- Test and Verify the Automated Traceability Results
- Architecture
- Design
- Developer Test
- Implementation
- Iteration Plan
- Project Plan
- Risk List
- Status Assessment
- Work Items List
- Actor
- Glossary
- Supporting Requirements
- Use Case
- Use Case Model
- Vision
- Test Case
- Test Log
- Test Script

Main Description

[New - Specific to: Automated Traceability]

When creating / instantiating any new artifact (documents, code, models, etc.) please follow these suggestions:

- Utilize words, acronyms and terms that are well defined in the project glossary, and make sure that you are using them in a consistent way.
- Create rich content that includes appropriate rationale and domain knowledge. This can substantially improve the results of automated traceability. For a more in depth explanation of this best practice refer to [HUL05]
- Make sure that all important elements (requirements, classes, tests, risks, iterations, stakeholders, actors, etc.) are uniquely identified, and that their textual information is complete and not terse.
- If you have documents that were created separately that use same terms with different meanings, you must use a tool that translates terms from one domain meaning to another.
- Construct meaningful hierarchies in your artifacts (such as headings within documents, or appropriate package names for code). This information is used by Poirot to strengthen candidate links.