EVALUATION AND SELECTION OF CASE TOOLS:

A METHODOLOGY AND A CASE STUDY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KORAY OKŞAR

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

FEBRUARY 2010

Approval of the Graduate School of Informatics

_____

Prof. Dr. Nazife Baykal

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Assist. Prof. Dr. Tuğba Taşkaya Temizel

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assist. Prof. Dr. Altan Koçyiğit

Supervisor

Examining Committee Members

Prof. Dr. Semih Bilgen	(METU, EEE)	_____
Assist. Prof. Dr. Altan Koçyiğit	(METU, II)	_____
Assoc. Prof. Dr. Onur Demirörs	(METU, II)	_____
Assist. Prof. Dr. Pekin Erhan Eren	(METU, II)	_____
Assist. Prof. Dr.Sevgi Özkan	(METU, II)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Koray OKŞAR

Signature : _____

ABSTRACT

EVALUATION AND SELECTION OF CASE TOOLS:

A METHODOLOGY AND A CASE STUDY

Okşar, Koray

M. S., Department of Information Systems

Supervisor: Assist. Prof. Dr. Altan Koçyiğit

February 2010, 224 pages

Today's Computer Aided Software Engineering (CASE) technology covers nearly all activities in software development ranging from requirement analysis to deployment. Organizations are evaluating CASE tool solutions to automate or ease their processes. While reducing human errors, these tools also increase control, visibility and auditability of the processes. However, to achieve these benefits, the right tool or tools should be selected for usage in the intended processes. This is not an easy task when the vast number of tools in the market is considered. Failure to select the right tool may impede project's progress besides causing economic loss. In this thesis study, a methodology is proposed for CASE tool evaluation and selection among various candidates and the points that separate this work from similar studies in the literature are explained. Moreover, the methodology is performed on a case study.

Keywords: CASE tool evaluation, CASE tool selection

ÖZ

CASE ARAÇLARININ DEĞERLENDİRİLMESİ VE SEÇİMİ:

BİR METODOLOJİ VE BİR ÖRNEK İNCELEMESİ

Okşar, Koray
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez yöneticisi: Assist. Prof. Dr. Altan Koçyiğit

Şubat 2010, 224 sayfa

Günümüz Bilgisayar Destekli Yazılım Mühendisliği (CASE) teknolojisi yazılım geliştirmede gereksinim analizinden dağıtıma kadar neredeyse her aktiviteyi kapsamaktadır. Organizasyonlar süreçlerini otomatikleştirmek ya da kolaylaştırmak amacıyla CASE aracı çözümlerini değerlendirmektedir. Bu araçlar insan kaynaklı hataları azaltırken süreçlerin kontrolünü, görünürlüğünü ve denetlenebilirliğini arttırmaktadır. Ne varki bu yararlara erişmek için, düşünülen süreçlerde kullanılmak üzere doğru araç ya da araçlar seçilmelidir. Piyasadaki araç sayısının fazlalığı göz önüne alınırsa bu kolay bir iş değildir. Doğru aracı şeçmede başarısızlık projenin gidişatına sekte vurmanın yanı sıra ekonomik kayba da sebep olur. Bu tez çalışmasında, birçok aday arasından CASE aracı değerlendirmesi ve seçimi için bir yöntemler dizisi sunulmuş ve bu çalışmayı literatürdeki benzer çalışmalardan ayıran noktalar açıklanmıştır. Ayrıca, yöntemler dizisi bir örnek olay çalışması üzerinde uygulanmıştır.

Anahtar kelimeler: CASE aracı değerlendirmesi, CASE aracı seçimi

ACKNOWLEDGMENTS

It is my privilege to express my sincere gratitude to my supervisor Assistant Professor Altan Koçyiğit for his guidance, stimulating ideas, criticism, encouragement and insight throughout the completion of this thesis.

I must thank my managers Suat Gümüşlüol, Özlem Değer, Seyhan Halisipek at OYAK Headquarters because of their generosity at giving me permission to leave for thesis meetings and their support for my case study. I should also thank my colleague Alper Aslan for his valuable comments and to my other colleagues because of their kind tolerance to my absences.

I am also grateful to my beloved one Emel Sarıkaya whose endless support encouraged me to proceed, to my sister Dilek and to everyone who have aided in one way or another.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHP: Analytical Hierarchy Process

CASE: Computer Aided Software Engineering

CI: Continuous Integration

COTS: Commercial off the Shelf

CRM: Customer Relationship Management

DSS: Decision Support System

ERP: Enterprise Resource Planning

HLC: High Level Criterion

IM: Instant Messaging

ISO: International Standards Organization

LLC: Low Level Criterion

MCDM: Multi Criteria Decision Making

RFP: Request for Proposal

RSS: Really Simple Syndication

SMS: Short Message Service

WAS: Weighted Average Sum

CHAPTER 1

INTRODUCTION

Software selection is a difficult process that tries to fulfill organizational requirements by evaluating and selecting a suitable package from alternatives that are present in the market. Several studies have been conducted in the literature that intends to solve this problem in various fields. This study focuses on software selection issues in the CASE tools area.

As defined in ISO/IEC 12207 a CASE tool is a software product that can assist software engineers by providing automated support for software life-cycle activities [24]. Besides providing automation, CASE technology today is also used to provide information about the software being developed.

The emergence of CASE tools dates back to 1980s when only a few basic tools existed that were working on specific fields. Today, we have a vast array of suppliers which generally provide more than one CASE tool and there are many open source tools in the market that are also in the competition. Today's CASE technology covers nearly all activities in the software development process ranging from requirements elicitation to maintenance.

To manage the increasing complexity of software development practices and technologies, many organizations are employing CASE tools in their processes. Although, usage of these CASE tools has been beneficial for the organizations, they didn't provide the magnitude of improvement predicted when they were first introduced.

This is generally linked to the creative nature of software development process which cannot be completely automated. Also, complex tools require a long learning time and some of them constrain the work of users which may lead to rejection or reluctant use. This makes selection of the right tool an important issue. The tools that lack the necessary functionality, the ones that have low usability or the ones that simply don't fit into the organizational culture may cause economic loss.

Success and approval of a CASE implementation in an organization may be improved by using a well designed CASE software evaluation and selection methodology. However, the CASE software selection task is usually performed under schedule pressure and a standard process for selection of a CASE tool is usually not defined. In case of a decision necessity, the decision makers may not have enough time to design the selection process in detail so they tend to use ad hoc approaches. If a comparison opportunity between tools exists, they tend to compare features which are easy or popular to measure. However, these features may be irrelevant or less relevant when the end user requirements are concerned.

The purpose of this thesis study is to provide a well defined and repeatable methodology for the organizations to adopt and apply in CASE software evaluation and selection. The proposed methodology employs a requirement driven approach for the purpose of eventually selecting a tool that is a "best fit" for the organizational practice that the tool will be used for. The methodology starts from defining the organizational requirements for CASE tool adoption and continues with progressive screening of the alternatives in the market. To help extracting requirement definitions, categorized question sets are presented. After screening phases, final candidates will be evaluated to measure how well each satisfies the predetermined criteria and then ranked using a well known ranking method in the field of multi-criteria decision making called Analytical Hierarchy Process (AHP) [10]. With the application of these activities, an inherently subjective process is tried to be formalized with the objective of increasing its accuracy.

Besides providing a tool selection process, the application of the proposed methodology also enhances the understanding of organizational requirements for CASE tools. Moreover, the literature search that will be made throughout the study and the findings that result from various calculation and comparisons will form a valuable knowledge base for future reference.

Several studies exist in the literature that covers software selection problem in general and a few of them investigate CASE tool selection in particular. However, these studies do not present an end to end methodology which covers criteria formation, evaluation technique and ranking method. Rather, they generally focus on presenting a set of criteria to be used in tool selection. Due to the diverse nature of software development organizations and a vast array of tool options in the market, definition of a standard criteria set seems difficult. Since each organizational software development formation has unique needs, the standard criteria set given would need extensive tailoring before they can be used. Moreover, new technologies are being added to the CASE tools area everyday which makes these criteria obsolete in a small time frame. The ISO 14102 standard for CASE tool evaluation and selection was also criticized from this perspective [6]. Recognizing this condition, this study does not present a set of predefined criteria for CASE tool selection. Rather, the focus is on definition of a methodology that covers criteria formation by the organization itself by eliciting the organizational requirements and examining the technology trends.

Another difficulty that exists for CASE tool selection is the high number of candidate tools. The CASE tool market is continuously growing both from the commercial and open source sides. New CASE tools are being introduced regularly which claim to possess the best functionality and characteristics.

The previously mentioned AHP method is considered suitable by this study for selection among CASE tool candidates. However, if we use it directly on all of the candidates and compare them with the multiple criteria of the organization, the comparisons and calculations may take enormous time.

Organizations generally cannot devote this much time to the selection process. Therefore, this study proposes two screening operations prior to ranking the alternatives. The aim of these operations is to reduce the number of candidates that will be subject to detailed evaluation. Usage of screening prior to ranking is also defended by Blanc and Korn [7]; however a method for screening was not given. To the best of our knowledge, the two stage screening method proposed in this study is not employed by any other study in this field.

Moreover, almost all the studies in the field mainly focus on selection of a single CASE tool that will be used in one particular field. However, one tool may not satisfy all the requirements of the intended field. Instead, a combination of several tools may need to be used and this is not a rare case. This possibility and its complications of this approach are also studied in detail in this thesis.

Furthermore, a case study is performed to demonstrate the application of the proposed methodology on the tool selection for one of the popular areas of agile development: continuous integration; an agile practice for which CASE tools are mostly used. Although the proposed methodology does not give any predefined criteria for software practices, this case study presents necessary criteria for the continuous integration area since this is the practice that the subject organization in the study is using the methodology for. Therefore practitioners may be inspired from this part of the thesis in case of tool selection for continuous integration as no specific work on this field exists in the literature.

This thesis consists of five chapters. The second chapter presents a summarized survey of literature on the topics concerned. In the third chapter, all the activities and steps of the proposed methodology are detailed in sequence. The fourth chapter presents the application of the methodology on a real case study and the last chapter presents the conclusion of the thesis.

CHAPTER 2

LITERATURE SURVEY

In this chapter, literature information concerning the areas investigated by this thesis is presented. In that respect, first section is devoted to the general software evaluation and selection problem. Second section provides information about the improvements and problems in CASE software today. Third section summarizes the studies performed specifically on CASE tool evaluation and selection. And the last section includes information about the continous integration practice of agile development which is the subject of the case study in chapter four.

2.1 SOFTWARE EVALUATION AND SELECTION IN GENERAL

The demand for software packages is continuously increasing. There are variety of packages being developed by the vendors to supply this demand. Also, open source software development community is growing and rapidly producing new software. The functionality of the packages offered varies from very simple to extensively complicated. These circumstances make the proper selection of a software package a difficult task. Selection of the wrong package may result in wrong strategic decisions with subsequent economic loss to the organization [8]. Therefore, better ways of decision making on this subject has been investigated by researchers.

Software evaluation and selection is an example of mutiple criteria decision making (MCDM) problems which involve making preference decisions over the available alternatives having multiple attributes. In such problems, the existence of limited resources generates the constraints and the value of the decision variables satisfying these constraints defines the feasible set. Then, each feasible solution is assigned a priority number reflecting the preferences of the decision maker. This number should be obtained with a criterion function.

According to Ballestero, only non-preferential technical information is required for the initial phase of this paradigm that is for defining the feasible set. In other words, the first phase defines what is possible from purely technical information. Actual preferences of the decison maker is reflected in the second phase that is when the criterion function is established. The intersection of both phases yields the feasible solutions which satisfy the constraints of the problem. The best or optimal solution is finally obtained by utilizing more or less sophisticated mathematical techniques considering multiple criteria [91].

Objectivity should be seeked when making decisions in such contexts however there are fundamental limitations for this according to Figueira. He points out these limitations as:

- The borderline between what is feasible and what is not is often fuzzy in real decision making contexts.

- Many data are uncertain, imprecise or ill-defined.

- In general, it is impossible to say that a decision is a good one or a bad one by referring only to a mathematical model . Organizational, pedagogical, and/or cultural apsects of the whole decison process which lead to making a decision also contribute to its quality and success [92].

In light of these difficulties, MCDM techiques proposed in the literature aims to help decison makers in:

- Sorting out alternatives that are superior among the studied set
- Ranking the alternatives in decreasing order of performance
- Choosing the best alternative

According to Jadhav and Sonar' s extensive research on the literature [8], the researchers have worked on evaluation and selection of software in the following fields:

- Commercial off the shelf (COTS) software
- CASE tools
- Simulation software
- Decision support system (DSS) software
- Analytical Hierarchy Process (AHP) software
- Knowledge management tools
- Data mining software
- Visual programming languages
- Enterprise Resource Planning (ERP) packages
- Customer Relationship Management (CRM) packages
- Expert system shells
- Operations management software

It is found that most of the research is made for COTS software evaluation and selection category. Jadhav and Sonar further classifies these studies according to their contribution to the field as the studies providing methodologies for software selection, the studies providing software evaluation techniques, the studies providing software evaluation criteria, and the ones that offer systems/tools to support decision makers in software selection. The results of this classification is presented in their paper [8].

2.2 CASE SOFTWARE

According to Sodhi's definition, "Computer-Aided Software Engineering (CASE) encompasses a collection of automated tools and methods that assist software engineering in the phases of the software development life cycle" [100]. Pressman, Sommervile, Forte and McCulley propose slighty varied definitions for CASE [101, 103, 102]. Combining these definitions, we can deduce simply that a CASE tool is a software component supporting a specific task in the software-production process.

CASE tools have been used by engineers since the early 80's where they were mainly performing computer aided documentation. They were being used to key in text and manipulate it by a visual interface. Later improvements led to the development of data dictionaries which stored details of all the data types and related processes. Computer aided diagramming tools were also developed to assist the software engineers and programmers to quickly draft and easily modify diagrams and designs. In later 80's, CASE tools that generated code were developed for various fourth generation programming languages [93].

Code generators were further enhanced in the 90's which enabled CASE tools to produce code in a more sophisticated manner from designs and data flows fed into the system. In these years, developing user friendly graphical user interfaces are emphasized. This has resulted in the greater involvement of the end users in software engineering process. Also, CASE tools that could be used in the entire life cycle of the software development paradigm were also introduced which includes project management tools and cost calculators. These tools made it possible to predict the resources and time scheduled of software in development [93].

High rates of CASE tool penetration into the market during these years is understandable because the total cost for human resources in software production was about $250 billion per year and even a modest increase in productivity would significantly reduce costs [2].

In the present, there are tools that support requirements elicitation and analysis, design and communication, enforcement of standards and methodologies, prototyping and rapid application development, reverse engineering and software maintenance [83]. Moreover, software engineering continues to become more complex with the availability of various software platforms for development. The requirement to seamlessly switch between different platforms has forced the CASE tools to evolve further to meet the needs of the industry. Software reengineering is also gaining importantance as a methodology and CASE tools have been developed addressing this area [93]. Tool vendors that seek to support a team in the software development process are trying to address such issues as team coordination and project management [96].

It seems that CASE technology will play a key role in the information technology market, and many new products will appear. However, this product proliferation and the richness of the functions offered are creating critical problems.

Besides increasing the tool capabilities, the additional features also increased the complexity of tools leading to steep learning curves. Kemerer states that the reason of limited usage of a CASE tool would be its complexity [111]. Iivari describes the same issue from a different perspective. He points out that the users' perception of the tool being complicated to work with is the main problem [112].

Moreover, assesing the capabilities of many products on the market is more difficult. Understanding their functional relationships between each other also requires careful examination since the terminology in this area is often misleading or confusing. According to Fuggetta's study terms such as tool, workbench, toolset, and environment are given very different meanings and interpretations among different organizations [2].

It is difficult, therefore, to develop a clear and systematic classification of the available technology for effective assessment and acquisition [2].

Several classification schemes are proposed in the literature to address this problem. Sommerville proposed a functional and an activity based classification where tools are classified according to their specific function or the process activities that they support [103]. Fuggetta proposed a classification acoording to the breadth of support that the tools offer. He grouped the CASE tools into three categories as tools, workbenches and environments in the order of increasing software development activity coverage [2]. However, according to Sommerville, the boundaries between these classes are blurred. It may therefore not always be easy to position a product using a classification. Nevertheless he states, a classification is a useful first step to help understand the extent of process support that a tool provides [103].

Furthermore, the need for integration in CASE technology is increasingly acknowledged by researchers and practitioners [2]. According to Thomas and Nejmeh, integration can be analyzed in four dimensions: data integration, control integration, presentation integration and process integration. Data integration deals with the management of information as a whole between the tools. Control integration deals with the combination of underlying environment functions according to project preferences. Presentation integration focuses on improving user interface interactions with the user and the process integration ensures that tools interact effectively in support of a defined process [104].

2.3 CASE SOFTWARE EVALUATION AND SELECTION

CASE tools can increase productivity in software development projects by supporting activities which are usually carried out with human efforts using little or no tool support. However, the introduction of a tool can also decrease productivity in some situations where much more effort should be spent to maintain the tool. This condition results in the need to make a detailed elaboration on tool evaluation and selection.

It has been found that the same tool can have very different effects on productivity depending on individual project characteristics. According to a study performed by Bruckhaus, Madhavji, Janssen and Henshaw, large productivity differences are observed when the same CASE tool is used in different sized projects and in the projects applying different development processes. Hence, they deduced a connection with the project size and tool productivity and conclude that a tool's performance may peak in a certain sized project. Also, one of their conclusions was that adopting a complex process can be substantially less expensive if the appropriate tools are also adopted [97].

According to the ISO standard on "Software Product Evaluation" (ISO 14598) [106], the evaluation process should promote four characteristics: repeatibility, reproducibility, impartiality and objectivity.

Moreover, according to Lundell and Lings, besides ensuring these characteristics, three primary dimensions need to be taken into account for an evaluation activity: the stakeholder dimension, the contextual dimension and the activity dimension [98].

When selecting stakeholders, a broad and representative selection is encouraged. Also, the roles' closeness to the usage context of the tool is stated as important. If multiple perspectives are involved in the selection process, validity of the results will be enhanced but effective and ongoing feedback is essential to create a sense of ownership. However, individual stakeholders can have very different goals considering the broad scope of the subject [98].

In such cases, Brown and Wallnau suggest that only aggreed, explicit and shared goals should be taken into account which can be treated as organizational goals [105].

Focus on the contextual dimension is very important because it is based on the realistic perception of an information system not only as a technical system but also as a dynamic social system [98]. This situation makes each organization unique in the business of software development. This is supported by the exploratory study conducted by Reeken and Trienekens. They investigated method and CASE-tool usage in sixteen largest organizations in Netherlands and one of their findings was the fundamentally different ways of usage of the same terms about CASE tools in these organizations [107].

This result shows the necessity to develop an evaluation framework within a particular organizational setting. Failure to consider this specific setting, can result in outcomes that have limited relevance for the organization [98].

The activity dimension of the evaluation is supported in the literature by different approaches and several activity sets are proposed but it should be pointed out that selecting or constructing a set of activities is very dependent on the contextual and stakeholder dimensions. However, failure to consider the activity dimension can result in poorly conducted evaluations which can produce unreliable outputs causing a lack of stakeholder confidence for the process [98].

At this point, it is important to state that evaluation and selection are different concepts. Evaluation is a process of measurement and assessment while selection is a process of applying thresholds and weights to evaluation results and arriving at decisions [99]. For the selection process, a selection algorithm is used. Among the mostly used ones in the literature are the simple weighted average sum (WAS) method and the structured "Analytical Hierarchy Process" (AHP) [4, 10]. Some fuzzy aproaches and various algorithms focusing on preference measurements and pairwise comparisons are also proposed. However, because of its suitability for qualitative decisions in a group setting, AHP method remains as the mostly used method in this area.

In 1992, Blanc and Korn studied CASE tool evaluation and selection while proposing a compact metholodogy. In their methodology, they stress the importance of screening which they call “formation of a short list” [7]. They also suggest matching user requirements with CASE tool capabilities instead of using a standard criteria set however they don’t offer a method for gathering the user requirements. Moreover, they don’t propose to collect technology information from external sources which is understandable when the narrow scope of the CASE tool sector is considered at the time of their research. They finally propose that the selection process can only arrive to a final conclusion when the users actually use the tool [7]. But this may not be possible in today’s business environment which increases the need of developing a more detailed methodology to aid in tool selection prior to actual use.

The ISO standard 14102 provides a set of activities and a structured charasteristic set for CASE tool selection [5]. These set of characteristics are extended from ISO/IEC 9126-1:2001 which defines the general model of software product quality characteristics [114]. ISO 14102 further categorizes these characteristics into four groups as: characteristics related to life cycle process functionality, CASE tool usage functionality, general quality and not related to quality. The objective of the proposed process is to obtain quantitative evaluation results upon which the final selection decisions can be made [5].

The standard is later criticized by Lundell and Lings in that the tool characteristics proposed are not complete and some have overlapping scopes. Although the lack of completeness was also acknowledged by the standard, Lundell and Lings state that the standard should have included method support for dealing with this lack. Also, they found the method proposed by the standard too qualitative and defend that it should include further scrutiny about human factors inherent in a specific organisational setting [6].

The learning aspect of the evaluation process is also stated in several studies. One of them is the study of Etzerodt and Madsen which emphasizes the adoption of such a learning perspective during the course of evaluation. They defend that the evaluation process is not measuring a fixed set of values but should be seen as a process where the actors involved learn from their own experiences [108].

Moreover, some studies in the literature focus on the sociological effects of the tool evaluation processes, especially when using pilot projects as an evaluation method. For instance, Sadler and Kitchenham point out that the effects resulting from the tool's capabilities should not be confused with the effects resulting from the reaction of users to the tools in such pilot projects. Because people may show unexpected reactions to the tools because of the novelty they bring into the environment [90].

It is important to state that the evalauation processes are not only proposed for assessing the tools prior to acquisition but also for the tools that are already in use. The studies covering this area mainly aim to reveal whether the tools' functions are sufficient for the job or being effectively used and whether the tool provides the expected benefits [13, 90, 110].

2.4 CONTINUOUS INTEGRATION

The proposed methodology for CASE tool evaluation and selection is detailed in chapter three and in the following chapter four, the application of this methodology on a CASE study is demonstrated. In this case study, the proposed methodology will be employed for an instution's CASE tool selection problem. The instution is in the endeavour of making their practices more agile and is looking for CASE tool solutions to be used in their software integration process. In other words, they intend to form a continuos intregration process with the help of CASE tools which will be determined by the proposed methodology. Therefore, a brief introduction to the continuous integration practice is given in this section.

Agile methods rely on an iterative approach to software development and are designed to support the development efforts for which the system requirements frequently change during the process. They are intended to deliver working software to customers as quickly as possible and they brought the concept of focusing on the software itself rather than on its design and documentation [103].

Continuous integration is a well known practice of the agile methods that is designed to accelerate software delivery thus enhancing project agility. To satisfy this goal, the practice involves mechanisms and activities to decrease integration times. Martin Fowler describes continuous integration as "a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily—leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible" [113].

Thus, continuous integration provides rapid feedback so that the state of the project can be learned several times a day. It reduces the time between when a defect is introduced and when it is fixed which leads to an increase in software quality [109].

A typical continuous integration scenario includes the steps detailed below:

1. A developer commits code to the version control repository. This repository is being checked regularly by the continuous integration server for changes.

2. Continuous integration server detects a commit to the version control repository. Then it gets the latest version to its environment (e.g. sandbox) and starts a build process by executing a predesignated build script.

3. After the build finishes, the continuous integration server generates feedback for the specified project members.

4. After notifications, continuous integration server continues to poll the repository for changes.

We can illustrate the components of a typical continuous integration system in the figure below (Figure 1) and descriptions for each component are given after the figure.

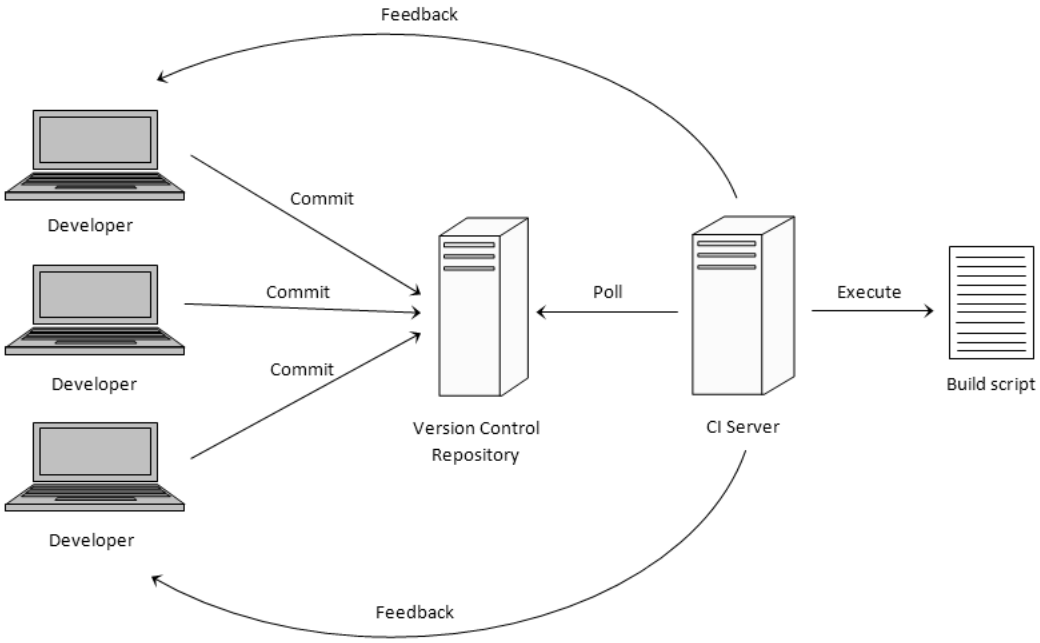

Figure 1: Continuous integration system components

- ***CONTINUOUS INTEGRATION SERVER***

The continuous integration server polls the version control repository and executes an integration build whenever a change is committed by a developer. When executing the build, it follows a predesignated build script which usually contains instructions about compilation, database integrations, running tests, running inspections and deployment information. The continuous integration server also provides feedback to interested parties about the result of the build.

- ***VERSION CONTROL SYSTEM***

Version control (also known as revision control or source control) is the management of changes to code files, executables, documents and other information stored as computer files. Automation and tool support is indispensable in this domain [94]. Changes are usually identified by a version number and kept in a database. However, the best solution for handling changes in software projects is the software configuration management process. This process covers additional practices to maintain software integrity and traceability besides the simple version control.

From the continuous integration perspective, the version control tool provides a "single source point" so that all source code is available from one primary location [109].

- ***BUILD TOOL***

In computer programming, compilation means translation of source code into executable code. A build on the other hand has a broader scope than compilation. It may include compilation, deployment, packaging, testing, inspection and other specific functionality. It provides a process for putting the source code together and verifying that the software works as a cohesive unit [109]. The build is described in a build script which is executed by a build tool. As stated before, the continuous integration server invokes the build tool when necessary.

- ***FEEDBACK SYSTEM***

In order for the continuous integration system send information concerning the builds, there should be a feedback mechanism in place. This mechanism is usually an e-mail application but can also be Really Simple Syndication (RSS), Instant Messaging (IM) or Short Message Service (SMS). The continuous integration server invokes the feedback system to send notification about the result of a finished build.

After setting up these components the automated continuous integration environment will be ready. However, automation of the continuous integration system is not enough to be completely safe from integration problems. Practicing routines like using a separate integration build machine, frequent commits to a version control repository and fixing broken builds immediately are also essential for performing an effective continuous integration [109].

CHAPTER 3

PROPOSED METHODOLOGY

FOR CASE SOFTWARE EVALUATION AND SELECTION

According to the literature survey made, a publication that addresses a complete methodology of software evaluation and selection does not exist. This is confirmed by the study of Jadhav and Sonar [8] which states that a publication that involves selection criteria, methodology, evaluation technique and practical application for the selection of CASE tools is not present.

The purpose of this thesis is to present a requirement driven methodology that covers all the phases of the CASE tool selection process including and giving emphasis to screening and evaluation criteria definition. The methodology is valid for the CASE tools that are developed to support a software engineering phase or practice and for the particular organization which forms the context of the evaluation and involves a software development group. It is defended that a tool cannot be best in all circumstances but can be best for a specific context that is a specific organization.

The methodology defined in this work will be referred throughout the thesis as "proposed methodology". It is formed by activities which in turn are formed by steps. The sequence of these activities and the artifacts they produce are shown in Figure 2.

As can be seen in the diagram, there are seven activities that form the methodology:

1. Initiation
2. High Level Evaluation Criteria Definition
3. Prescreening
4. Low Level Evaluation Criteria Definition
5. Screening
6. Evaluation and Comparison
7. Ranking and Selection

The activities are mainly sequential however the “Prescreening” activity and the “Low Level Evaluation Criteria Definition” activity may be performed in parallel. But both of these activities should be finished in order to pass to the “Screening” activity.

After completing the “Ranking and Selection” activity, a CASE tool or several CASE tools are selected and will be put into use in organization’s processes. This phase which is defined as “Implementation and Maintenance” is not in the scope of the proposed methodology but may be addressed in future research.

It is important to note that after applying the proposed methodology, no tool may pass the screening phases. This situation means that none of the tools in the current CASE market satisfies the organization’s criteria. In such a case, the criteria may be reconsidered by the evaluators. After the reconsideration, if the evaluators are able to relax the limiting criteria in such a way that their constraining effect is reduced to allow some CASE tools to pass, then the selection process may be repeated. Otherwise, the overall acquisition process may be abandoned or options like developing an in-house tool from scratch or modifying an existing tool may be considered.

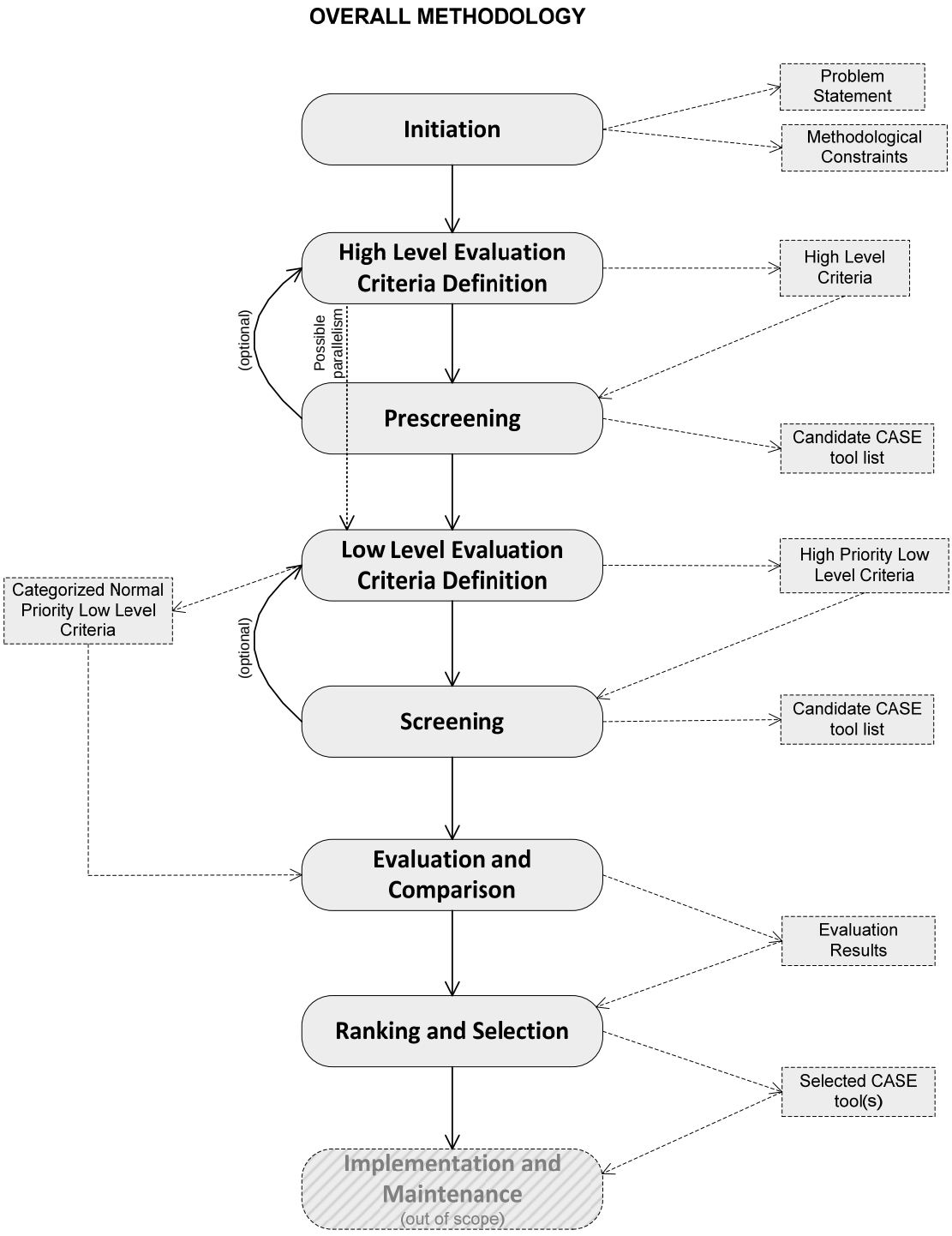

Figure 2: The proposed methodology

The proposed methodology starts with the initiation activity in which the rationale behind the CASE tool acquisition effort is clarified. Also, organizational commitment to the process is assessed and an evaluation team is formed in this activity.

In the high level evaluation criteria definition activity, the aim is to extract the most important and obvious requirements of the organization. These requirements usually arose from organizational constraints and previous tool experiences in the organization. Also in this activity, it is determined whether multiple CASE tool combinations will be evaluated or a single CASE tool solution is sufficient.

In the following prescreening activity, the first candidate CASE tools are selected according to the high level criteria formed in the previous activity. It is possible that none of the tools that exist in the market pass the prescreening phase. In that case, the high level criteria obtained in the previous step may optionally be relaxed to a degree that some tools can satisfy them or the acquisition process may be abandoned.

The low level evaluation criteria definition activity may start in parallel with the prescreening activity or after it. In this activity, a more detailed organizational requirement analysis is performed combined with an external analysis. The results are combined to form the low level criteria which will then be prioritized and categorized.

The criteria which are identified as high priority are utilized in the screening activity where the organization further eliminates some of the tools. After finishing this last screening, the final candidate list will be ready for detailed evaluation. Again, it is possible that no tool be able to pass the screening in which case an optional return to the previous criteria determination activity may be made to relax some of the criteria.

The two screening phases utilize high priority criteria which are generally in the form of pass/fail clauses that do not require extensive measurements to assess. In the following evaluation and comparison activity, the capabilities of the candidate tools are assessed and compared against the normal priority low level criteria which require more detailed examination and research.

In the final ranking and selection activity, results of the previous comparisons and weight determination techniques presented by the AHP method are utilized to find the final ranking of the tools. Then the recommended tool or tools to be used in the organization can be disclosed. These tools may optionally be studied in a pilot project before the implementation if the necessary resources are available for such a work. The detailed explanations of these activities are given in the rest of this chapter.

It is important to note that although the flow of the activities and steps is clearly defined in the methodology, the time and effort that will be dedicated to them is not rigid and can be adjusted according to the needs.

3.1 The Initiation Activity

In this very first activity of the proposed methodology, we aim to clarify the reasons behind the attempted CASE tool acquisition and whether the organization has fully committed on the process. This activity is divided into four sequential steps:

1. Rationale Determination
2. Commitment Determination
3. Methodological Constraints Determination
4. Evaluation Team Formation

Flow of these steps is demonstrated in the figure below. (Figure 3) Explanations are given for each of the steps.

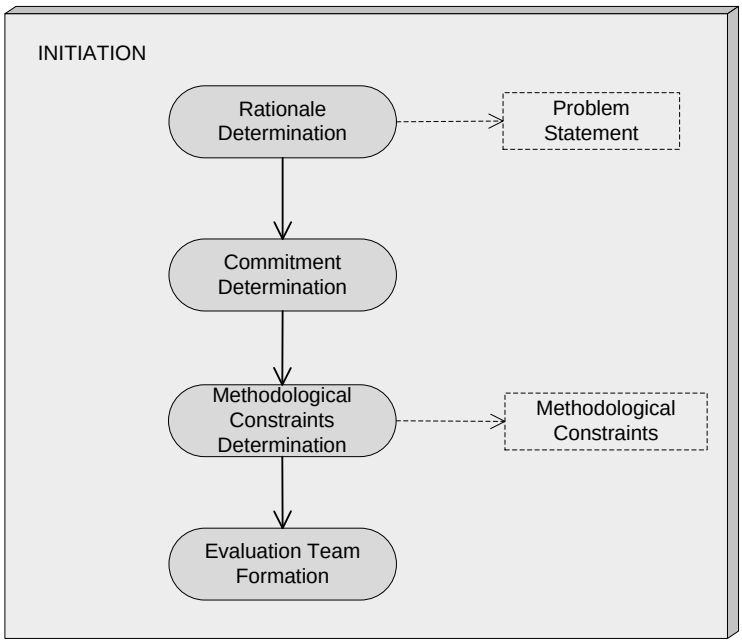

Figure 3: Breakdown of the Initiation Activity

3.1.1 Step1: Rationale Determination

Before entering the process of CASE tool evaluation and selection, the organization should set the rationale for CASE tool acquisition first. Rationale forms the reason that leads the organization to CASE tool acquisition. It may be one of these:

- The organization may want to automate a complete software development phase or a part of it

  The organization may be conducting one of its development practices or phases manually. This manual execution may be leading to some human related errors or may be consuming valuable time and resources. In this scenario, the organization wants to automate this set of human controlled practices by using CASE software. It is important to note that the organization may already be using CASE software for some parts of this practice but the CASE tool or tools being used may not cover the practice as a whole. In this respect, the organization may utilize the proposed methodology to find a suitable tool or tools that cover the practice as a whole. During the process, the existing CASE tools may be replaced with betters which is the case given below in the second possible rationale.

- The organization may want to replace a CASE tool with a better alternative

  In this case, the organization may be having problems or difficulties with the CASE software that they are currently using. Or, in a complete automation process like the case above, the existing tools may not have enough functionality or suitability for integration with the other tools.

As a start, the properties of the existing CASE tool(s) should be examined. Properties which the organization benefits from and the ones that cause problems for the teams working with the tool should be identified. Also, significant functionality which is required but not present in the tool should be identified. For example, in the case of a configuration management tool, the benefited functionality may be the tool's web interface, the drawback of the tool may be its poor performance and an additional desired functionality may be a supported integration with the organization's integrated development environment. This information will be input to the following criteria definition activities.

- The organization may want to change their software development methodology or part of it and in need for a CASE tool that will support the new methodology

  In this case, the organization is having problems not with its current toolset but with its current methodology. Thus, the organization is in the process of modification of its development practices and the new practices will be supported by CASE tools.

- The organization may just want to enter a CASE tool evaluation process for forming a knowledge base for future decisions.

  In this case, the development or management team in the organization anticipates that a CASE tool acquisition may be required in the future. This anticipation may be due to a possible change of development methodology or according to the observed trends of the CASE tool technology. The areas of software development that the organization is interested in for CASE tool adoption will be evaluated and the proposed methodology will be applied to find the suitable candidates.

This proposed methodology is designed to be used in one of the circumstances described above. If it is intended to be used for another reason, the activities forming the methodology may not be suitable or they need to be modified.

After the clarification of the rationale, a brief problem statement describing the issues and their sources that lead to CASE tool acquisition should be prepared. It will act as a concise description of the problem and a starting point in the progress of solution forming. A good problem statement should include the definition of the problem, the reason behind the problem and the expected solution. It can also include the information that explains why the organization orients towards an acquisition process instead of writing a tool itself.

3.1.2 Step2: Commitment Determination

Before entering the processes of CASE tool evaluation, selection, acquisition and implementation, organizational commitment should be ensured. This commitment is expected both from the management who will financially support and supervise the acquisition and from majority of the potential users of the system to be acquired. If the organization lacks commitment from either source, the acquisition and implementation process may be in jeopardy and all of the efforts may be wasted.

The short term and long term economic benefits of using the intended CASE technology may need to be discussed with management to get their approval for the process. Also, if the actual users (e.g., developers) of the system are reluctant for the implementation, achievement of the expected benefits of the CASE tool(s) is unlikely. User related benefits and potential improvement that can be gained after the learning period of the system may be explained to the users to get approval from them.

Also, since the CASE tool(s) will require installation and maintenance support from system administrators, it is better that they and the other staff who will indirectly be involved in the process be consulted and their opinions taken into consideration.

Moreover, these mentioned stakeholders should also agree on using a well defined methodology like the one proposed with this study for the CASE tool selection process. Usage of such a methodology will minimize the risk of selecting a wrong tool which will cause economical loss and demoralization of the team. Also, the evaluation process will form a knowledge base for the organization in case of future evaluation and selection efforts. However, it requires time and staff to perform and these resources may not be readily available. In this respect, the proposed methodology is prepared to finish in a reasonable time by utilizing progressive screening to minimize the number of CASE tool candidates to be input to the evaluation and ranking activities which take longer times.

The questions below should be answered in order to determine commitment to the process. The answers should be positive in order to continue to the methodology. One exception to this is the second question. The answer maybe "no" to this question if and only if the organization has made an intentional decision to only use a freeware CASE tool.

Commitment Requirements Question Set:

1. Has the management fully agreed on acquisition and implementation of a CASE tool?
2. Has the management fully agreed on funding a CASE tool?
3. Has the management fully agreed on supporting the selection process (the proposed methodology)?
4. Has the development team agreed on acquisition and implementation of a CASE tool?

3.1.3 Step3: Methodological Constraints Determination

In this step of the initiation activity, the organization should determine its constraints about the application of the proposed methodology. In this step, the organization has already agreed on the usage of the methodology however the constraints on it like the amount of resources that can be devoted have not yet been set.

The methodology involves browsing the literature, measurements for the evaluation activities and group meetings for decision making. The detail level of all these activities may be adjusted according to the available resources which will be determined in this step. However, it should be noted that the more resources dedicated to the process, the more accurate result will be obtained and the more extensive knowledge base will be formed.

The questions below should be answered in order to determine the methodological constraints.

Methodological Constraints Question Set:

1. What is the number of personnel that can be delegated as the evaluators?
2. How much time can the evaluators allocate for the methodology work?
3. How much time can the organization devote to the selection process totally?

After determining the methodological constraints, the organization may form the evaluator team in the next step.

3.1.4 Step4: Evaluation Team Formation:

The activities in the methodology should be performed by an evaluation team formed by organization employees. It is suggested that the team include a coordinator who is skilled in the domain of the intended practice for CASE tool usage and minimum of two evaluators. The evaluators will gather the organizational requirements, convert them to organizational criteria, screen and evaluate the tools according to these criteria. The coordinator will manage these activities and arrange group meetings for prioritization and other decision making activities.

More than one evaluator is required in order to conduct search and information gathering activities in parallel during the application of the methodology. It is also suggested that the team members be representatives of the intended tool user group but not be biased towards a specific tool. They should be able to dedicate time for the activities and organize evaluation meetings with other employees when necessary. Besides the evaluation team, other stakeholders like the developers or the management will also be consulted when gathering organizational requirements.

After forming the evaluation team, the initiation activity is considered to be finished and the next activity of the proposed methodology namely “The high level evaluation criteria definition activity” can start.

3.2 The High Level Evaluation Criteria Definition Activity

In this activity, the high level evaluation criteria that represent the organization's primary needs will be extracted.

The organizational constraints limiting the selection and attributes coming from the experiences with the existing CASE tools in the organization will form the high level requirements of the organization. It is important to note that the requirements that will originate from this activity should be "high level" that is they should be obvious without needing a measurement or extensive searching. Requirements such as "the tool's price should be lower than $10000" falls into this category. Then these high level requirements will be expressed as the high level criteria for tool selection.

The steps that form this activity are given below:

1. Constraints Determination
2. Existing Toolset Examination (optional)
3. Tool Area to Search Determination
4. High Level Criteria Determination

Methodological constraints were determined in the previous activity since they apply to the selection process and not to the tool(s). In the constraints determination step of this activity, the organization should determine its constraints about the tool(s). Then the existing set of tools and experiences with them should be investigated to aid tool search and replacement decisions. However, this step is only valid if tools exist in the organization. Otherwise, this step may be passed. The step called "Tool Area to Search Determination" is for determining the number of the tool areas that the intended practice spans. In other words, this is where the organization tries to assess whether the methodology will be performed to select a single tool or a combination of tools.

After the tool areas are determined, high level criteria can be finalized in the last step called “High Level Criteria Determination”. The flow of these steps and the artifacts they produce are depicted in the figure below. (Figure 4)

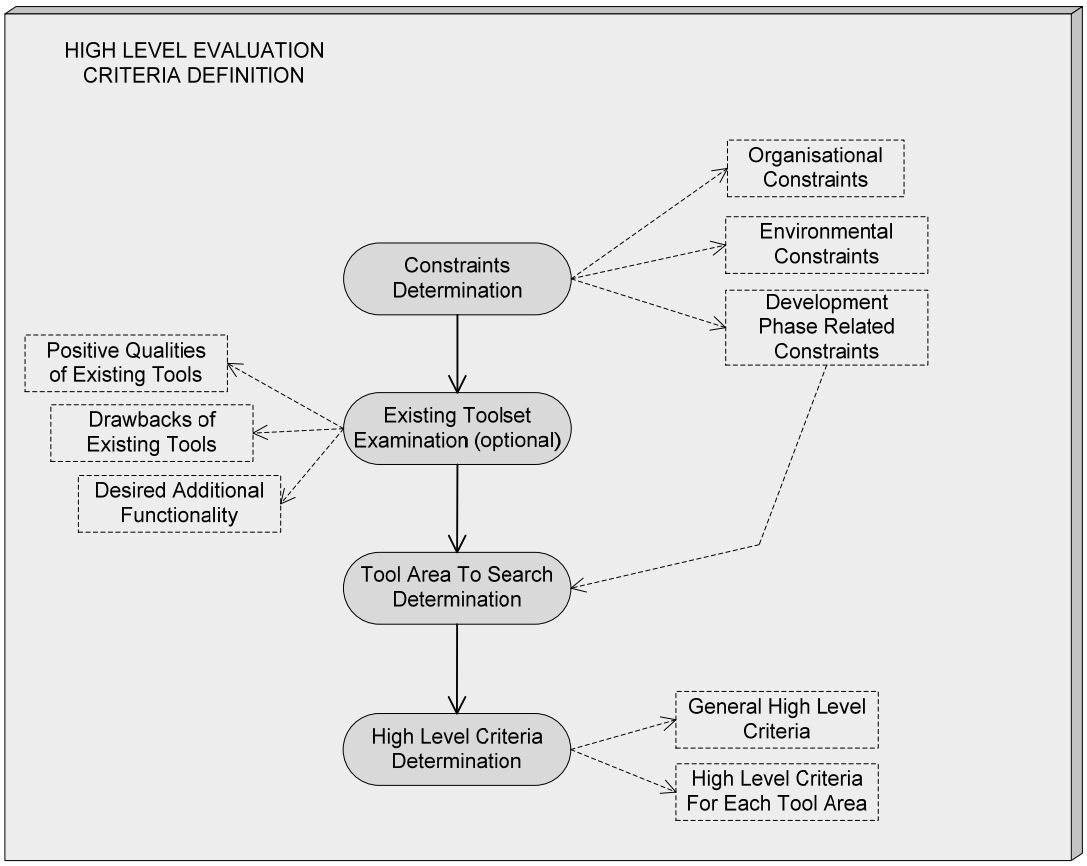

Figure 4: Breakdown of the High Level Evaluation Criteria Definition Activity

3.2.1 Step1: Constraints Determination

After determining the rationale for acquisition in the previous activity, the organization should now set their constraints concerning the CASE tool(s). It is important to note that a constraint does not represent a desirable feature of the tool but represents a “must” feature or characteristic that the tool should absolutely possess.

The constraints will be expressed in three categories:

- Organizational constraints
- Environmental constraints
- Development phase related constraints

The explanations of these constraints are presented below.

3.2.1.1 Organizational Constraints

This type of constraints usually originates from the resources of the organization that can be devoted for the acquisition. Also constraints like mandatory conformance to an international or an organizational standard falls into this category. The list of organizational constraints can be extended by the organization that is applying this methodology. However, the evaluators in the department should be careful about the item they are adding in that it should really represent a constraint for the selection. In other words, it should be something that the tool must absolutely satisfy the requirement of it. The questions below should be answered as a starting point.

Organizational Constraints Question Set:

1. *What is the maximum expenditure we can spend on the tool?*

   Note: Answer to this question may be zero if only freeware solutions will be considered

   Note: It is better to also discuss the preferred amount of expenditure at this point so it can be treated as a further requirement.

Note: If multiple tool areas are discovered in the third step of this activity, total of the costs of tools belonging to these areas should be taken into account.
Note: Support costs of the tools should also be taken into account when searching their price.

2. *What is the number of licenses needed?*

Note: There are three types of commercial licenses usually found in the market for CASE tools called floating, node-locked and authorized licenses. Floating licenses are served from a license pool upon request and don't have a computer or user restriction. Node-locked licenses are restricted to be used in a specific computer and authorized licenses are restricted to be used by a specific user. The exact number of licenses required will be determined by examining the license type provided by the tool, the density of usage of the tool and the number of workers that will use the tool. Here, only a rough estimation about the license numbers required in case of these different licensing schemes should be given. This number will be useful when evaluating tools according to their cost.

3. *How much time should be required to fully integrate the tool into the organization's development practices?*

Note: This amount is not related to the time needed to apply the selection methodology. Rather, it is the time needed for implementation of the tool after it has been selected which concerns factors like the ease of installation of the tool or the tool not having a steep learning curve (easy to learn, easy to implement)

4. *Should the tool conform to an international standard that is adopted in the organization?*

   Note: The organization may be bound to conform to a general standard concerning its services or a practice specific standard like MIL-STD-973 for configuration management. In such a case, the tool that will be acquired should also conform to this standard.

3.2.1.2 Environmental Constraints

This type of constraints originates from the organization's hardware and software environment or specific technology compliance mandates.

The organization policies or practices may restrict disk usage, memory usage or may require specific chipset support. These necessities form the hardware related environmental constraints. Only the candidates that satisfy these requirements will be evaluated.

Software related environmental requirements include operating system support requirements and interoperability or compatibility requirements with other software being used in the organization such as the database system or an application server. Only the CASE tools that work on the specific operating system or the ones that are compatible with the specific software of the organization will be evaluated further.

Moreover, the requirement for the tool to support a software development technology or framework which the organization is using forms a specific technology compliance constraint. This type of constraints should also be assessed when evaluating tools.

The questions below should be answered to proceed with the methodology. The organization may add additional environmental constraints in the same lines.

Environmental Constraints Question Set:

1. *Do we need a tool that supports a specific processor architecture?*
   (E.g.: SPARC, X86, X64, etc…)

2. *Do we need a tool that supports a specific CPU bus size?*
   (E.g.: 32 or 64 bit systems)

3. *Do we have a constraint on the CPU usage of the tool?*
   (E.g.: The tool's CPU requirement should be dual core 2 GHz at most)

4. *Do we have a constraint on the disk usage of the tool?*

   Note: This includes the installation size plus the estimated growing size.

5. *Do we have a constraint on the memory usage of the tool?*
   (E.g.: The tool's memory consumption should be around 256 MB and up to 512 MB as the maximum)

6. *Do we have a constraint on the network usage of the tool?*
   (E.g.: The tool should work without a network connection or the tool's continuous bandwidth usage should be around 1 kb/sec up to 5 kb/sec as the maximum)

7. *Do we need a tool that supports a specific operating system for the server?*
   (E.g.: Windows Server 2003, Red Hat Enterprise Linux 5.0, etc…)

8. *Do we need a tool that supports a specific operating system for the clients?*
   (E.g.: Windows XP SP3, Windows Vista, Open Sues v11, MacOsX 10.6, etc…)

9. *Do we need a tool that should integrate with a specific software being used in the organization?*

   Note: Here, integrability is used in the same sense as interoperability meaning that the two systems should be able to exchange information between each other and work together
   (E.g.: The version control tool should integrate with the build tool being used in the organization)

10. *Do we need a tool that should be compatible with a specific software being used in the organization?*

    Note: Here, compatibility is used as the tools ability to use the services of an underlying platform
    Note: The operating system compatibility will not be covered here since it is covered with another question because of its importance
    (E.g.: The Java based CASE tool should be compatible with JVM 5.0 or the CASE tools web interface should be compatible with Safari browsers)

11. *Do we need a tool that should be compatible with a specific software development technology being used in the organization?*

    (E.g.: The CASE tool should be developed for use in JAVA/J2EE platforms)

3.2.1.3 Development Phase Related Constraints

The organization will be using the CASE tool(s) for a software development phase or practice. Thus the CASE tool(s) should be covering this phase or practice. This forms a main constraint on the tool: its intended area to work. We certainly are not interested in the tools covering unrelated areas.

In this step, the organization should elaborate on the phase or practice that the CASE tool(s) will be used. It should be investigated whether the practice involves clear-cut subpractices. Results of this examination will be input to the "Tool Area to Search Determination" activity in which the concerned development practices or subpractices will be matched to CASE tool areas.

The questions below should be answered to extract the development phase related constraints.

Development Phase Related Constraints Question Set:

1. *In which software development phase or practice will the CASE tool be used?*

2. *To which subpractices can the practice be divided?*

   Note: Answer to this question will not reveal a constraint actually but will be used in the "Tool Area to Search Determination" activity.

After determining the "Development Phase Related Constraints" the constraint determination step will be over and the evaluation team can proceed to the next step which is "Existing Toolset Examination".

3.2.2 Step2: Existing Toolset Examination (optional)

This step is only valid if there are CASE tools being currently used for the intended practices in the organization. It can be skipped if there are no existing CASE tools. Because of this reason the activity is “optional”.

In this step, we intend to extract the positive and negative qualities of the existing tools so that we can decide on keeping them or replacing them with a better alternative. At the end of this step the organization should produce three outputs namely: “Positive Qualities of Existing Tools”, “Drawbacks of Existing Tools” and “Desired Additional Functionality”. These outputs will be used when preparing the high level criteria and will aid to determine what to look for when searching a better tool.

The source of this information will be the users of the existing tools. The gathering of the information may be accomplished by using a survey which consists of the questions below or by using a group meeting.

Existing Toolset Examination Question Set:

1. *Which tools are currently being used in the organization for the intended field?*

2. *For each of these tools, what are the functionalities or good quality characteristics that the users like and benefit from?*

3. *For each of these tools, what are the drawbacks?*

4. *For each of these tools, what are the desired additional functionalities?*

   Note: The desired functionality may include the positive replications of the drawbacks.

After gathering the answers, they should be documented in three categories: Positive Qualities of Existing Tools, Drawbacks of Existing Tools and Desired Additional Functionality. At the end of this step, it can be concluded that the existing tools will absolutely be kept and no evaluation corresponding to their areas is needed. If this is the case, the to be acquired tools' support for the integration with the existing tool(s) should be included as a high level criterion.

3.2.3 Step3: Tool Area to Search Determination

If a complete software development phase or practice will be automated, then the overall practice and the subpractices forming it should be evaluated to match the existing CASE tool areas in today's CASE tool technology. According to this examination, the solution may comprise more than one CASE tool area (the technology area or software development practice for which CASE tools are developed in the market) so more than one CASE tool. Then some parts of this methodology should be repeated for each CASE tool area.

The possible subdivisions of the intended practice were identified in the constraint determination step when determining development phase related constraints. This information will be utilized here.

This step of the methodology involves browsing the literature for CASE tool areas that correspond to the intended practice as a whole, its each subpractice and possible combinations of its subpractices.

Therefore, we can state that; the tool areas are the selected elements of the set of subpractice combinations of the intended practice. If a software development practice "X" can be divided into three distinct subpractices like A, B and C then the CASE tool area possibilities will be: ABC, AB, AC, BC, A, B, C.

One or more of these possibilities should be selected according to the examination of the literature. For example, if we assume that the practice which will be handled by CASE tool(s) is “configuration management”. From the constraint determination step, we have the information that the configuration management practice can be divided into four subpractices: configuration identification, configuration control, configuration status accounting and configuration auditing. Now, in the current step, we should determine the tool areas that will be investigated. Information sources like the internet should be searched to find whether CASE products are available for the configuration management as a whole or for its subpractices. Such a result may yield from that examination:

The CASE tool areas that will be taken into consideration are:

1. Configuration management tools (tools that include all the functionality)
2. Configuration (version) control tools

This result means that the organization couldn’t find any tools for configuration identification, configuration status accounting or configuration auditing. The tools in today’s technology either include all the functionality of the configuration management practice or just the configuration control subpractice. So this gives us two areas for further consideration. Here however, the second area is a subset of the first area which means that if it is revealed that the organization has criteria that the second area tools do not address, it will be eliminated and the method will continue on one area only.

Another point that should be considered in this step is the scope which the organization will use when examining the tool areas. That is, if a software practice is divided into three tool areas, will the organization examine a tool corresponding to a tool area just from the perspective of its benefit to the whole area or with the other properties it have?

For example, if we assume that we intend to acquire CASE tools for a software practice that spans three tool areas as in the previous example as A, B and C. When we are investigating tools for the A tool area, if we find tools that also cover an additional D area, (that is tools belonging to the AD area), will we consider only the functionality of A or will we consider the functionality of D as well? If the organization is also interested in the functionality of D area, then one of the tool areas to search will be AD instead of only A even if functionality of D is not related for the intended practice (ABC).

Another case is comparing a tool area combination with an identical single tool area. For example, if we assume that we have tool areas AB, A and B. At the end of the methodology, we would find a best tool for AB, a best tool for A and a best tool for B. In such a case, we should determine whether we will acquire both of the tools of A and B (assuming that they are interoperable) or we will acquire the single tool of AB. If we assume that we call the combination of the capabilities of the tools belonging to A and B as X and we call the capability set of the tool of AB as Y. Then for this decision, it is required to make a final comparison between X and Y to conclude about the tool or tools that will be acquired.

After determining the tool areas, the evaluator team can continue to the last step of this activity which is “High Level Criteria Determination”.

3.2.4 Step4: High Level Criteria Determination

When the evaluator team finishes all previous steps, it will have requirements originating from constraints and from existing tool experiences. Also, the tool areas to consider will be already determined before starting this step. In this step, these requirements will be consolidated and converted to high level criteria for tool selection.

Therefore, this is essentially a step of consolidation and conversion. The conversion stated here is the conversion between requirements to criteria since they are not equivalent concepts. Despite some other work on the field of software selection which does not differentiate between organizational requirements and tool criteria, the proposed methodology emphasizes this difference in agreement with the SEI process guide on COTS software selection [9].

The main difference between requirements and criteria is that the requirements are usually formed to express needs however criteria should express the expected capabilities. This in turn changes the structure of the criteria in such a way that they should be more concrete than high level requirements. They should be easily assessable and defined in clauses expressing capability needs. The evaluation of these clauses should not require measurement and should be answerable in a yes/no fashion.

Also, as stated in the activity's title, the criteria prepared in this activity should be high level that is they should represent the broad outcomes the business requires, rather than the specific functions being expected. One exception can be the criteria derived from the negative tool experiences of the workers. These criteria can be more specific than the other criteria originating from the constraints however they are also addressed in this activity as high level criteria.

In that respect, the proposed methodology offers the below format of clauses to use when defining criteria:

- The CASE tool(s) shall support…
- The CASE tool(s) shall be able to…
- The CASE tool(s) shall include…
- The CASE tool(s) shall cost…
- The CASE tool(s) shall be developed for…

For example, a requirement of the organization which originates from negative tool experiences of the developers would be: “The system shall be easy to configure” which is too abstract and does not point to a tool capability. In the current step of the methodology, this can be converted to a tool criterion such as: “The CASE tool shall include a graphical wizard for the X functionality” which is concrete enough to assess. Moreover, it can be seen that the criteria clauses do not include negatives such as the “tool shall not include”. If such negative requirements exist, they should be converted to their positive restatements when deriving criteria from them.

Criteria originating from constraints generally are converted to criteria that are applicable to all of the tool areas. Therefore these are consolidated into “General High Level Criteria” output. General criteria apply to all the tool(s) that will be acquired. However, there may be some exceptions for some criteria which should be noted when declaring them. Remaining criteria will be organized for each of the tool areas found in the previous step and will be consolidated into “High Level Criteria for Each Tool Area” output.

After categorizing the criteria formed here as general criteria and criteria for each tool area, the criteria should be numbered. The numbering should start with the HLC prefix meaning High Level Criteria for example HLC1, HLC2, etc…

When the numbering ends, the evaluation team is considered to have finished this step and may continue to the prescreening activity.

Although the evaluation team is considered fully responsible for the process, it may be useful at this step to arrange a group meeting with all of the stakeholders of the system to discuss the formed criteria and gain everyone’s approval.

3.3 The Prescreening Activity

Numerous tools have been developed in the market supporting nearly every aspect of software development. In order to decrease the evaluation effort that will be needed in the subsequent phases of the methodology, we need to narrow the number of candidate CASE tools. The publications in the software selection area generally focuses on the ranking and selection algorithms used for selecting a tool for defined candidates. However, if we decrease the number of candidates in the first place, we will greatly reduce the resources we spend on applying the methodology. At the end of this activity, tools that don't address the organization's primary needs will be eliminated from consideration. The high level criteria we obtained from the previous step will be used for this objective.

It is important to note that the evaluation team can also start the "Low Level Evaluation Criteria Definition" activity at the same time it starts the prescreening activity as depicted in Figure 2. Since the two activities are independent, the evaluator team may split and perform the activities in parallel to increase the pace of the selection process.

The prescreening activity is composed of two steps which are:

- Tool information gathering for each tool area
- Criteria matching for each tool area

Schematic representation of the activity is given in the figure below. (Figure 5)

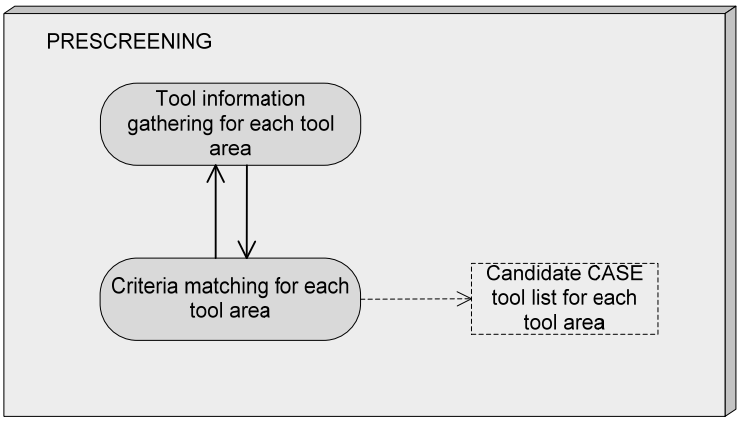

Figure 5: Breakdown of the Prescreening Activity

The questions below should be answered at the end of this activity:

1. What CASE tools are available in the market for the specified tool area? (corresponding to a software development practice)
2. Which CASE tools in the previous set satisfy the primary criteria established in the high level evaluation criteria definition phase?

As can be seen in the figure the steps that form the activity can be sequentially performed however it is suggested to conduct them in parallel giving feedback to each other. This is because the aim of this activity is not gathering extensive information about the tool but rather making the screening as fast as possible. The suggested procedure to apply the steps is given at the end of the step definitions.

3.3.1 Step1: Tool Information Gathering for Each Tool Area

In this step, the organization should start searching the information sources to determine the candidate tools for each tool area. For instance, if the tool area is UML modeling, all the tools that are developed for UML modeling should be documented. It is better to organize the tool information by using tables. The information to be collected is given below. If the organization decides to use tables to organize data, suggested columns are also given in the table. Each row in this case will represent a tool candidate as can be seen in the example. (Figure 6)

For each tool area:

Table 1: Tool information table format (column-wise)

Column1 : Tool Name	The tool's name
Column2 : Producer-Vendor	The tool's producer if it is commercial or the team that first developed the tool if it is freeware-open source
Column3 : License	Can be commercial if the tool is being sold with a price (proprietary) or can include the type of free software license
Column4 : Tool Site	The tool's website
Column5 : Tool Version	The version of the tool that will be evaluated during screening and the following activities of the methodology

Example:

CI Server tool information table				
Tool Name	**Producer-Vendor**	**License**	**Tool Site**	**Version**
CruiseControl	Originally developed by ThoughtWorks	BSD-style license	[25]	2.8.2
Hudson	Originally developed by a worker in Sun	MIT license	[26]	1.341
Continuum	Apache Software Foundation	Apache 2.0 license	[27]	1.2.3
Luntbuild	Javaforge	Public Domain	[28]	1.6.3

Figure 6: Example from the case study

The fastest information source to get this data is the internet. The searching process may be conducted separately for each tool area in parallel. This is an important step in the overall methodology since a tool that exists but missed from consideration can be the tool that fits best in the organization. Also, adding a new tool in the later stages of the methodology means repeated screening, evaluation and computation efforts.

3.3.2 Step2: Criteria Matching for Each Tool Area:

In this step, the organization should start evaluating tools according to the high level criteria. It is better to start with criteria that can be found most easily like an environmental criterion such as “The CASE tool shall support RedHat Enterprise Linux 5 operating system for their server software.” This kind of information can easily be found in the tool’s web pages. Other possible information sources include tool brochures, existing tool users, forums and user groups, etc…

For criteria matching, usage of tables is recommended. Each row of the table will represent a candidate CASE tool and each column will represent a high level criteria number. A check mark is written at the cell corresponding to a criterion-tool match (meaning that the tool satisfies the criterion). The tool(s) that satisfy all the criteria pass to the next stage of the methodology.

An example of this kind of checklist is given below. (Figure 7)

Configuration management tool – criterion matching table (checklist)													
	HLC ID												
Tool Name	**14**	**20**	**3**	**6**	**10**	**8**	**7**	**1**	**16**	**5**	**4**	**12**	**2**
AccuRev	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BitKeeper	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
ClearCase	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Synergy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Co-Op	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne	ne	ne	ne
Perforce	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PureCM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 7: Example from the case study

When a tool fails from one criterion, the other criteria do not need to be examined since failure to satisfy a high level criterion is enough for rejection. This is the reason of the "NE" marks in the example meaning "not evaluated". For future reference, the failure reasons of the rejected tools may be documented under the checklist in a "notes" section. If none of the tools can pass the criteria, evaluators may decide to revise them to make a less constraining criteria set or abandon the process.

The matching process may also be conducted separately for each tool area in parallel therefore increasing the overall efficiency of the methodology.

It is stated that the two steps forming the prescreening activity may be performed in parallel. Below is the suggested procedure for this case.

Suggested Procedure to Apply the Steps:

1. *Construct the tool information gathering table for a specific tool area.*

2. *Search the tools that are developed for that tool area and write them sequentially on the rows of the table.*

3. *Rank the criteria from most easily findable to least easily findable.*

4. *Construct the criteria matching table by putting the most easily findable criterion at the first column and then putting others next to it according to the previous ranking.*

5. *Start gathering tool information for the first tool in the tool information table and go on with the others.*

6. *When searching information for the tool information table, if you encounter information regarding the first few criteria, mark them immediately in the criteria matching table.*

With this procedure, it is intended to decrease the total information search time compared to the sequential application of the steps.

At the end of this step, a subset of CASE tools that represent the first candidates for selection should be obtained and the evaluator team can proceed to the next activity. However, if all the tools are eliminated during this step, a return to the previous high level criteria determination activity can be made for relaxing some of the criteria or the process as a whole may be canceled if such a relaxation is not acceptable.

3.4 The Low Level Evaluation Criteria Definition Activity

After determining the high level evaluation criteria, it is time to define the organization's low level evaluation criteria which constitutes of the detailed functional and other types of requirements being expected from the tool(s).

At the end of this activity, low level organizational criteria for the CASE tool will be formed which will include two groups: "normal priority low level criteria" and "high priority low level criteria".

Then, the normal priority criteria will be subdivided into three categories: functional requirements, quality requirements and supplier/community requirements. Nonexistence of low priority criteria may be questioned here but it is found unlikely that an organization would rate its criteria's priority as low. Moreover, the normal priority criteria's importance may be adjusted in the ranking activity. The current activity's real purpose is to separate the high priority criteria from the rest.

The low level evaluation criteria definition activity is composed of 4 steps which are given below. The first two of these steps should be performed in parallel if more than evaluator exists.

- Organizational requirement analysis for each tool area
- External analysis for each tool area
- Criteria formation
- Criteria prioritization and categorization

Schematic representation of the activity flow is given in the figure below. (Figure 8)

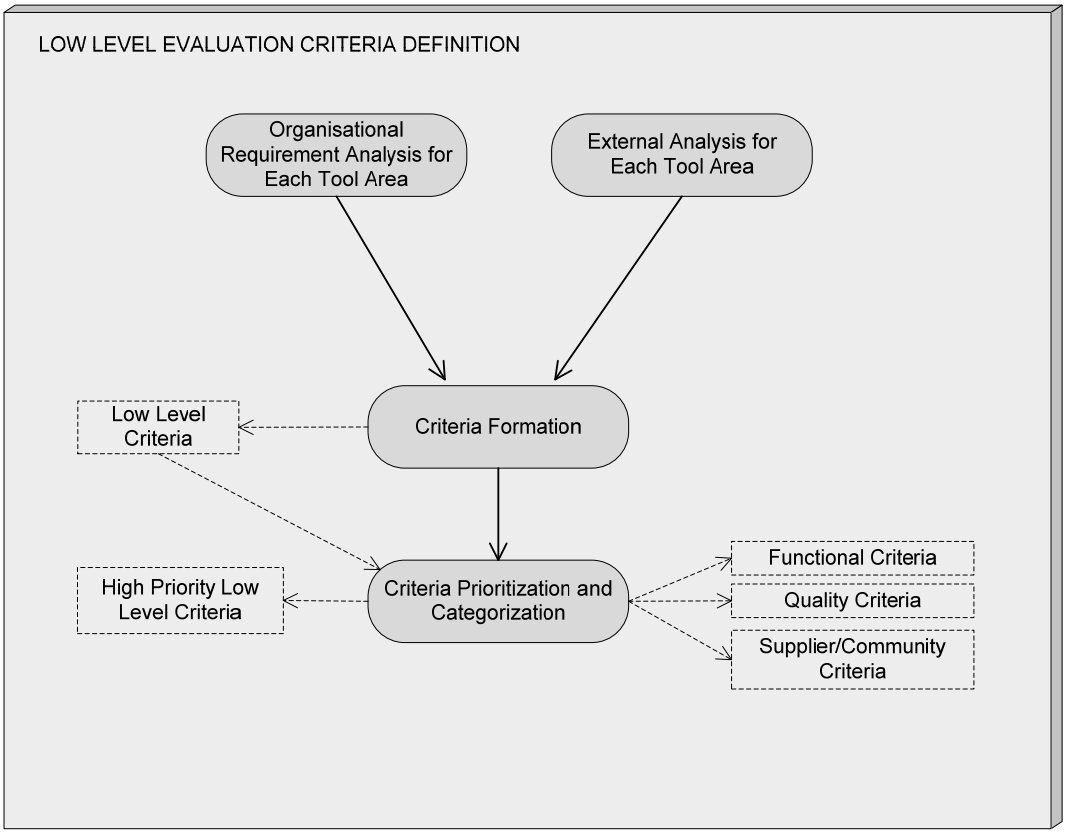

Figure 8: Breakdown of the Low Level Evaluation Criteria Definition Activity

The process starts with the activity of "Organizational Requirement Analysis" in which the evaluators collect in-house requirements. An external analysis is made in parallel to obtain additional possible requirements. In the following "Criteria Formation" activity, these requirements which are obtained from different sources (external and internal sources) are combined excluding the duplicates. Also, in this step the conflicts between the requirements are found if there are any. These conflicts should be resolved by negotiations between the sources of the conflicting requirements and the evaluators.

In the “Criteria Prioritization and Categorization” step, the criteria are prioritized to distinguish the high priority ones and the remaining normal priority criteria are distributed into three groups according to their types as functional, quality or supplier/community.

3.4.1 Step1: Organizational Requirement Analysis for Each Tool Area

In this phase, functional and nonfunctional requirements of the organization are identified as comprehensively as possible. The source of the requirements may be potential tool users, the organizational policies, other employees who have experience for the CASE tools in the intended field and may also include the managers of the departments in which the tool will be used. Also, hardware and software maintenance engineers and system administrators should also be considered as a source. Some of these are stakeholders who won’t use the system themselves but may impose some constraints (thus forming some requirements) on it. Interviews, group meetings or surveys can be used to gather data. In case of survey usage, the experience level of the respondents and their attention contributes to the consistency and reliability of the results [83]. For getting more details for the data gathering activities, the reader is referred to the “requirements elicitation” resources in the literature.

Some of the guidelines that apply to requirements engineering are also beneficial when gathering organizational CASE tool requirements such as;

- The requirements should be unambiguous and verifiable. (a template should be used to decrease ambiguity)

- The requirements should be complete. (all services required by the user should be defined)

- The requirements should be consistent. (requirements should not have contradictory or overlapping definitions)

- The requirements should be realistic. (potential users may have unrealistic demands according to current technology)

Additionally, below points may hold for the CASE tool area:

- Managers may have demands that increase their control but make the end user's (e.g. developers) work difficult.

- End users may have subjective demands that only aids in his/her working style but not beneficial or required by others.

Nonfunctional requirements (quality, supplier/community) which relate to the system as a whole should also be considered since they may be more important than functional requirements because of the fact that functional requirements may have workarounds but nonfunctional requirements like performance may not.

Moreover, it is better to have requirements which are quantitatively defined and can objectively be tested. Obviously some requirements like maintainability cannot be quantified though. The organizational requirement analysis step can be performed in parallel with the external analysis step given below.

3.4.2 Step2: External Analysis for Each Tool Area

In this step, low level requirements are gathered from outside of the organization. Sources include tool web pages, independent online resources like comparisons, tool brochures, tool presentations or demonstrations, independent consultants and several publications that are available in the literature for CASE tools.

As another information source, ISO 14102 [5] standard presents a classification of CASE tools according to the lifecycle processes they support. The key characteristics that should be present in the CASE tools supporting these areas are defined structurally. Besides categorizing according to lifecycle processes, the standard also presents general characteristics of a CASE tool organized into three groups: CASE tool usage functionality, general quality and not related to quality. Some of these characteristics may be used as a low level criterion.

Another source of low level criteria may be the online or published documentation of the CASE tools. Most CASE tools offer online information centers from which extensive information about the tool can be obtained. Examining the documentation of candidate tools may reveal their strengths and weaknesses against each other and may lead to formation of some additional requirements. However, it is important to note that the online documentation of a tool may stress the tool's strengths but not include its weak points. Moreover, if the forums will also be considered, they should be examined skeptically because the information people make would be biased. They could be reporting positive qualities of a tool that they used to and defaming another because of their lack of knowledge.

Apart from the tool's functionalities, information about the vendor that produces the tool may also be needed. For searching company profiles and products they offer, ICP (http://www.icpcredit.com/) may be used. Also several other sources exist for this purpose.

Moreover, CASE tool distributors may be contacted for a presentation of their offerings. They may provide tool brochures or working demonstrations of the tool. These kinds of meetings may reveal additional needs or desired functionality of the organization for the tool. However, it is better to invite different vendors that are competitors in the same area to gain different perspectives and to make unbiased decisions.

3.4.3 Step3: Criteria Formation

After finishing both of the previous steps, the organization should have requirements collected from internal and external sources. In this step of the methodology, the organization should consolidate the requirements that are elicited for each tool area.

This step is necessary because the evaluator making the organizational requirement analysis and literature analysis may not be the same person. As stated before, this is also an advised condition for the two activities to proceed in parallel which induces a decrease in the time required to complete the methodology.

Therefore, there may be overlapping or conflicting requirements. In this step, these inconsistencies will be tried to be eliminated.

Also in this step, the previously gathered requirements should be converted to criteria and documented using a standard template. During this conversion, a requirement may be detailed into two or more criteria according to the requirement's abstraction level. Moreover, negative requirements should be converted to their positive restatements in this step.

Writing all the organizational criteria according to a standard template reduces ambiguity, increases the effectiveness of the organization of the criteria and also decreases the errors that may result when the criteria definitions are left to the evaluators' own interpretations. This template should include a rationale section which details the reasons behind the criterion that will help decision makers in the evaluation phase. The template below can be used for this purpose. Writing the definition and rationale sections in a detailed manner is encouraged to prevent the possibility of misunderstanding since the criteria may be defined and evaluated by different workers.

Evaluation Criteria definition template:

Table 2: Evaluation criterion table format (row-wise)

Item name	**Description**
ID	The criterion identifier
Title	The title of the criterion
Definition	The explanation of what is required by this criterion. This section may involve several clauses formatted in the criterion pattern which represent the breakdown of the goal of the criterion.
Rationale	The explanation of the reason behind this criterion, in other words; this is the area we describe why we need this criterion.
Source	The information about where we have found this criterion from.
Adaptability	The information about whether this criterion is adaptable that is whether it can be changed or worked around to achieve the same or similar output
Priority	(Optional)
Type	(Optional)

The priority and type of the criterion in the template are left as optional as they will be documented in the next step however for data integrity they can also be evaluated and documented when constructing evaluation criteria tables.

3.4.4 Step4: Criteria Prioritization and Categorization

In this step, all the criteria that are formed should be prioritized in each tool area. The priority will be either normal or high. The high priority low level criteria will be input to the next phase of the methodology which is the screening phase. In order a criterion to be high priority it should resemble a characteristic of the tool that should be present as a "must" when the organizational requirements are concerned. A normal priority on the other hand represents the desire for a criterion but not absolute necessity. Moreover, high priority criteria represent characteristics which the tools may possess at the required level or may not possess at this level. A pass or a fail decision is made according to this information. However, in the case of a normal priority criterion, the candidate tool will be assessed according to how well it satisfies this criterion and take a score about this, not just a pass or a fail judgment.

After prioritizing the criteria, the next step is categorizing the normal priority criteria into three groups which will make up the type of the criteria: functional criteria, quality criteria or supplier/community criteria. The high priority low level criteria need not to be categorized.

Functional criteria: This type of criteria describes what the system should accomplish. In other words, they resemble a capability that should be offered by the CASE tool.

Quality criteria: These are the criteria that are not directly related to the individual system functions but they relate to the system as a whole. They are criteria like: efficiency, reliability, portability, etc… Organizational criteria that are only valid for the specific organization that is in the process of CASE tool acquisition are also considered in this category. For instance, a criterion that states that the CASE tool operations shall support ISO 27001 is a quality criterion.

Supplier/community criteria: This type of criteria describes the quality of support that the vendor of the tool provides. For instance, a criterion such as the tool supplier should provide on-site support in the first month of CASE tool usage period is a supplier criterion. Also, the criteria about the tool's user community fall under this category.

It is important to note that these are the types that are sufficient to categorize the criteria according to the methodology however they can be divided into finer categories or new types can be added by the user. One of the candidate types for decomposition is quality for instance. However, since the functionality is the most important factor in the CASE domain, this decomposition was not decided to be necessary in the proposed methodology. The execution of the methodology does not change upon addition of new types but the user should be cautious about adding types that are clearly defined and do not overlap.

Below template may be used to document the criteria priorities and types:

Table 3: Evaluation criterion priority and type table format (column-wise)

Item name	**Description**
Column 1: Criterion ID	The criterion identifier
Column 2: Priority	The criticality level of this criterion (should be "normal" or "high")
Column 3: Type	The information about whether this criterion is a functional criterion, quality criterion or supplier/community criterion.

An example of this table from the case study section is given below for quick reference. (Figure 9)

CI server criteria prioritization and categorization results

Criterion ID	Priority	Type
LLC1	High	NA
LLC2	Normal	Functional
LLC3	Normal	Quality
LLC4	High	NA
LLC5	High	NA
LLC6	Normal	Supplier/Community

Figure 9: Example from the case study

After this step, high priority criteria which will be used for screening should be ready and normal priority criteria should have been categorized according to their type. So the screening activity can start.

It was stated at the end of the high level criteria determination activity that a group meeting with the stakeholders may be arranged for validation purposes. This condition also applies here so that the formed low level criteria and their priorities may be demonstrated to the stakeholders of the system to gain their approval as a way of validating the process.

3.5 The Screening Activity

After determining all of the criteria, the screening phase of the methodology can start. At the beginning of this activity, the tool information that has been gathered in the previous prescreening activity and the high priority low level criteria should be at hand. In this activity, more CASE tools that don't satisfy the needs defined as high priority low level criteria will be screened. This is the last screening phase in the proposed methodology and the tools that pass this screening enter the evaluation phase for detailed examination. This phase is executed in two steps:

- Tool information gathering for each tool area
- Candidate CASE tool selection for each tool area

Schematic representation of the activity flow is given in the figure below. (Figure 10)

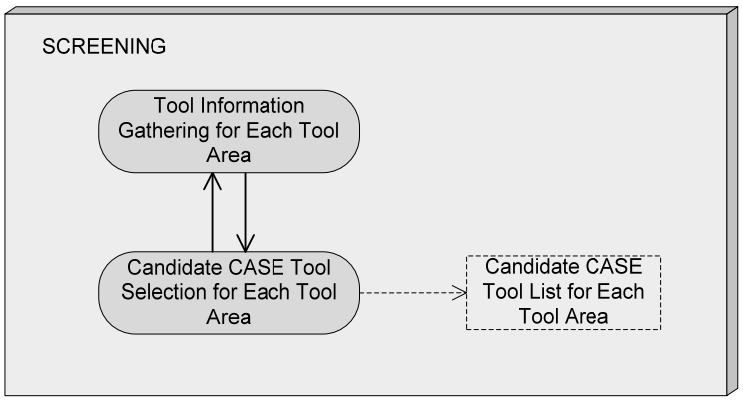

Figure 10: Breakdown of the Screening Activity

This activity in fact is very similar to the prescreening activity. The only difference is that in this activity, the screening of the tools is performed against low level criteria which are more detailed and require more searching in order to assess. As in the case of prescreening, the two steps forming the activity are performed in parallel.

3.5.1 Step1: Tool Information Gathering for Each Tool Area

Some tool information was gathered in the previous prescreening phase however this information may not include data that entails whether the tool satisfies the high priority criteria found in the previous activity. So to check this, additional tool information may need to be gathered. The tools' own documentation may provide sufficient information for this step. There is no format suggested to organize the tool information gathered in this step. As mentioned before, this step should be followed in parallel with the step below.

3.5.2 Step2: Candidate CASE Tool Selection for Each Tool Area

In this step, a checklist similar to the one constructed for the prescreening activity is prepared that shows whether the tools being examined actually satisfy the elicited high priority low level criteria. The tools that fail to satisfy or offer a workaround for a single high priority low level criterion is enough to be eliminated. It is suggested that the reason for their elimination is documented under the checklist in a notes section for future reference. At the end of this phase, a final list of tool candidates that passed from all of the screening procedures is obtained. This candidate list will be input to the evaluation and comparison activity. However, failure of the all the candidates during this step is also a possibility. In such a case, a return to the previous low level criteria determination step may be performed to relax some of the criteria in order to allow some candidates to pass.

3.6 The Evaluation and Comparison Activity

In the beginning of this activity, the evaluators should have determined the tools that satisfy all the high level and low level criteria that have high priority. All the tools that were able to pass up to this stage can be used in organization's intended practice. However, one of them should be selected for each tool area. For this purpose, the tools are evaluated against the normal priority low level criteria to determine the one that is a best fit when the needs of the organization are considered.

This activity is composed of two sequential steps:

- Assessment method determination for each criterion
- Assessment and comparison for each criterion

These steps should be performed for each tool area but each tool area's evaluation can be made in parallel.

Schematic representation of the activity flow is given in the figure below. (Figure 11)

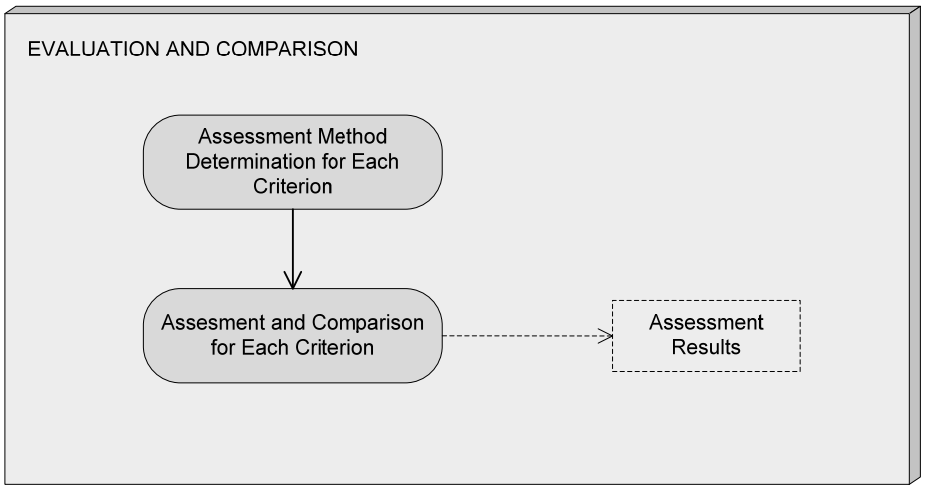

Figure 11: Breakdown of the Evaluation and Comparison Activity

3.6.1 Step1: Assessment Method Determination for Each Criterion

To conduct the evaluation process, assessment method for each tool criterion should be determined. Possible assessment methods are given below.

- *Official tool documentation lookup:* Commercial tools have dedicated sites that contain tool documentation. Freeware tools may also have official websites maintained by the group which promotes the tool. The documentation that comes with the tool installation is also considered as official in this study.

The technical details about the tools' functionalities can usually be found in the tool documentation. Also some quality metrics about the tool may be included and can be assessed from the documentation. For the information that exists in the documentation but the evaluators want to see themselves, the following method can be used.

- *Case study on tool (evaluation copy):* Hands-on inspection on the tool is one of the most reliable methods [3]. Most of the CASE tool vendors offer directly downloadable or upon-request evaluation copies. Some of them restrict tool functionality or scalability, some of them restrict license period and some of them restrict all. In spite of these limitations, most of an organization's criteria can be tested on an evaluation copy. Of course, freeware tools can be downloaded for this purpose with no limitation. If an evaluation copy does not exist for a specific tool, the vendor can be contacted for a permitted work on the real copy or this method is not used.

  Some quality criteria like ease of installation, easy to use GUIs or easy configuration can best be assessed by working hands-on with the tool. After downloading the evaluation or real copy of the tool, a case study over a scenario that is close to reality can be performed to observe the tool behavior and assess the subject criteria. If the vendor of the tool does not provide an evaluation copy or testing the subject criterion requires extensive product knowledge, then the method below can be used.

- *Demonstration or information request from the supplier:* The tool suppliers are usually generous about providing information or making a demonstration of their tools. In case of a criterion that is hard to test for evaluators but easy to show by the vendor can be assessed using this method.

Local distributors of the firms can be found from their web sites and contacted. In case of a demonstration request, the evaluators should concentrate on questioning the features of the tool which are the subjects of the criteria since the vendor team would exaggerate their tool's strong points and give information about unrelated areas.

Also, a request for proposal (RFP) can be submitted to the vendor to get information about very comprehensive and detailed criteria but it should be taken into account that preparing an RFP is a very time consuming task [7]. If a distributor of the vendor does not exist in the region of the organization, then evaluators may ask about the subject criteria via email. However this situation also raises concerns about the expected tool support during the tool's lifecycle in the organization if it is chosen to be used.

In case of freeware tools, this method would be not applicable however there are some firms which offer commercial support services for open source tools. If such a firm can be found, they can be contacted in regard of this method.

- *Visiting an existing user group:* In this method, the organization arranges a meeting with another organization that is preferably doing similar business and has been using the subject tool for a considerable time. This visit may also be arranged by the supplier of the tool since they tend to promote their success stories.

  This method can reveal first hand information about some tool characteristics which the evaluators cannot realize using another method. Robustness is an example for such kind of a characteristic although an idea can be grasped looking at the tool's defect count or testing an evaluation copy.

- *Online tool newsgroup or forum search:* Online search can also be regarded as an assessment method however it should be utilized when all the other methods are not available or as a secondary method after utilizing one of the other methods. This is because the information that can be obtained from these sources may lead to incorrect decisions since the source of the information is usually not reliable.

  When using this method, it is better to look for general newsgroups or forums that focus on the tool area rather than a specific tool.

- *Tool reference search:* This method involves an investigation of the vendor's references. In other words, the evaluators try to find which organizations are clients of the vendor in regard of the subject tool. A large number of clients (high market share) and tool's long presence in the market are good signs of tool reputation. Also, a large number of clients that are similar in size and conducting similar business with the organization can be desired. Some firms like Gartner and Ovum regularly assess products in this manner [87, 88].

Selection of a proper assessment method is a matter of available time and facilities. The above method list is presented as a guide only in that another method that can yield the desired result can also be utilized by the organization. It is also important to note that these methods can better be applied in combination to increase the accuracy of the results.

The template below is proposed to place the information gathered in this step.

For each tool area:

Table 4: Criterion-Assessment method table format (column-wise)

Item name	Description
Column 1: Criterion ID	The number of the criterion
Column 2: Criterion Name	The name of the criterion
Column 3: Assessment method	The assessment method of the criterion. This can be one of the methods given above or a combination of them. In case of a combination the order defines precedence.

An example of this table from the case study section is given below for quick reference. (Figure 12)

Criterion – Assessment method table for CI server tool area

Criterion ID	Criterion Name	Assessment Method
LLC2	Extension mechanisms	Official tool documentation lookup
LLC3	Robust working	Visiting an existing user group Official tool documentation lookup Online tool newsgroup or forum search
LLC6	Vendor and tool reputation	Tool reference search
LLC9	Ease of installation	Case study on tool evaluation copy

Figure 12: Example from the case study

After determining the assessment methods for each criterion in each tool area, the evaluators may pass to the next step in the methodology.

3.6.2 Step 2: Assessment and Comparison for Each Criterion

In this step, the evaluators should gather all the stakeholders and decision makers for the system to perform the actual assessments for each criterion using the methods defined previously. The group decision making activity would offer many benefits including improved overall decision quality and decision making effectiveness. However, forming the group from people who have different perspectives is important since people having the same role tend to weight the priorities similarly. For instance, management would weight managerial factors heavier than non-management members [84].

The results should be documented using a tabular format including a quantified comparison.

"Fundamental verbal scale for pairwise comparison" that is developed by Saaty in his "Analytical Hierarchy Process" (AHP) is used for quantifying the comparison (Table 5) [10]. Here, only the pairwise comparison of the candidate tools is performed according to AHP. The results will be used in one of the steps of the following ranking and selection activity where the full form of AHP will be utilized.

Table 5: Saaty's "Fundamental Verbal Scale for Pairwise Comparison" [10]

Expressed Judgment of Preference	Numerical Value
Extremely preferred	9
Between very strongly and extremely	8
Very strongly preferred	7
Between strongly and very strongly	6
Strongly preferred	5
Between moderately and strongly	4
Moderately preferred	3
Between equally and moderately	2
Equally preferred	1

Fundamental verbal scale for pairwise comparison is used when comparing two alternatives against a criterion. The procedure involves constructing a tabular matrix where rows and columns are filled with the exact alternatives. Then, the alternative in the first row is compared to each alternative in the columns. If the alternative in the row is better compared to the alternative in a column, a value selected from the verbal scale (Table 5) according the amount of superiority of the better alternative over the other is inserted in the corresponding cell. If the alternative is worse than the alternative in a column, reverse values are used. Value of one is used if the two alternative are the same or perform same according to the subject criterion. Therefore the left diagonal values are always one. This is illustrated in the example below.

Suppose we have three alternatives named as Alternative A, Alternative B and Alternative C. According to a criterion X:

- Alternative A is very strongly preferred over Alternative B.
- Alternative A is moderately preferred over Alternative C.
- Alternative C is strongly preferred over Alternative B.

In this case, the comparison table for the criterion will be like the one given in table 6 below.

Table 6: Comparison table example

Comparison	**Alternative A**	**Alternative B**	**Alternative C**
Alternative A	1	7	3
Alternative B	1/7	1	1/5
Alternative C	1/3	5	1

To summarize, in this step, the evaluations using the proper assessment method(s) defined in the previous step and pairwise comparisons according to AHP verbal scale are made. The template below is proposed to consolidate the findings and comparisons. (Table 7) It is prepared according to the case of three candidates and can easily be extended by the user to accommodate more. (By adding a row and a column for each)

For each criterion:

Table 7: Evaluation and comparison table format (row-wise)

Item Name	**Description**		
Criterion ID	The criterion identifier		
Criterion Name	The name of the criterion		
Candidate 1	Details of assessment findings of candidate 1 concerning this criterion		
Candidate 2	Details of assessment findings of candidate 2 concerning this criterion		
Candidate 3	Details of assessment findings of candidate 3 concerning this criterion		
Comparison	**Candidate 1**	**Candidate 2**	**Candidate 3**
Candidate 1	1	Result of verbal scale comparison	Result of verbal scale comparison
Candidate 2	Result of verbal scale comparison	1	Result of verbal scale comparison
Candidate 3	Result of verbal scale comparison	Result of verbal scale comparison	1

An example from the case study is given below. (Figure 13)

Evaluation and comparison table for LLC20				
Criterion ID	**LLC20**			
Criterion Name	**Graphical build trends**			
CruiseControl	The Cruise Control's dashboard interface provides visualization of build results. The department couldn't find any graphical build trend representation in standalone CruiseControl however this functionality is available through various plugins. Configuration of dashboard should be done via an XML file.			
Hudson	Hudson also includes a dashboard interface in which several graphical representations are offered. However trend graphics are mainly for test tool integrations and test results. Build result graphics are comparably limited.			
QuickBuild	QuickBuild includes statistical graphs that can show build and test trends.			
TeamCity	TeamCity offers custom graphic creation functionality which allows the user to create a graphical representation of the build data. This is a strong point in that the user can create any type of chart however the configuration should be made via an XML file.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	1/3	1/7	1/6
Hudson	3	1	1/6	1/4
QuickBuild	7	6	1	2
TeamCity	6	4	1/2	1

Figure 13: Example from the case study

After finishing the assessment for all of the criteria, this activity is considered to be finished.

3.7 The Ranking and Selection Activity

Since all the evaluations are made, this last activity of the methodology merely involves some calculations and a final assessment to find the best candidate for each tool area. The activity is composed of two steps.

- AHP application
- Final selection

The first step includes six sequential sub-steps which can be seen in the figure below. (Figure 14) After completing these steps, a ranking is formed upon which the final tool selections can be made. The details of the steps are given next.

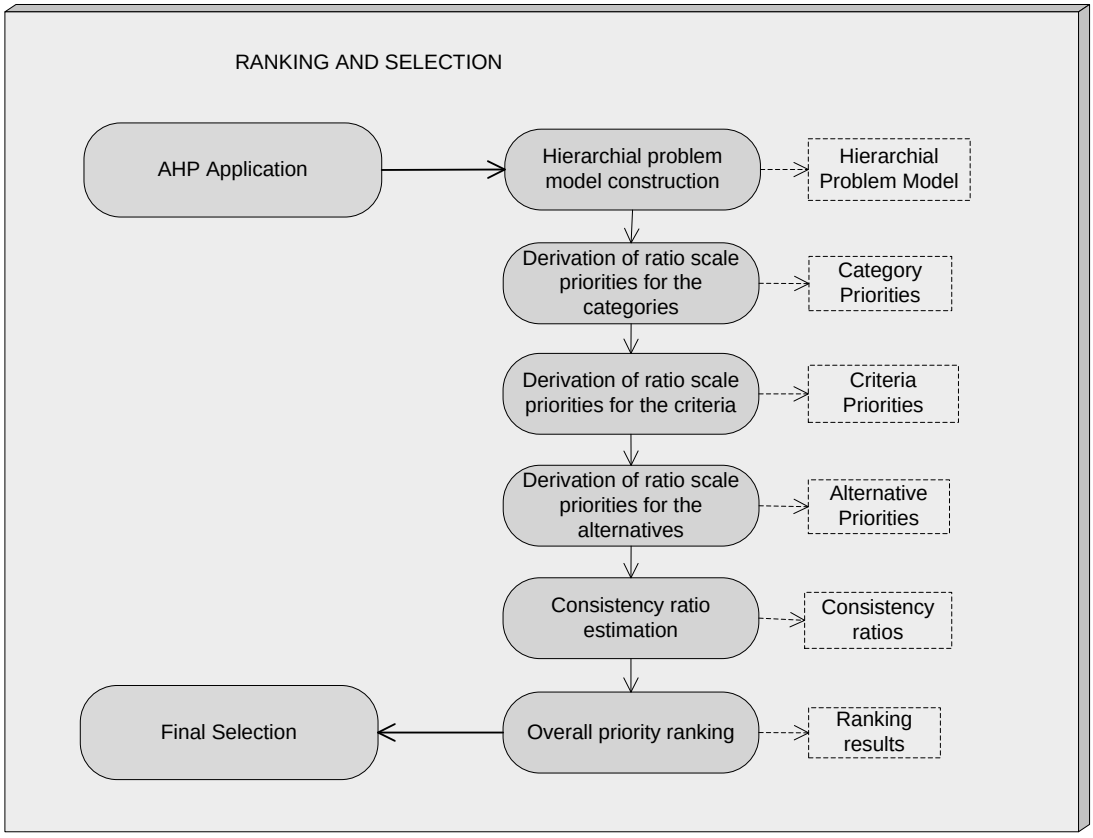

Figure 14: Breakdown of the Ranking and Selection Activity

3.7.1 Step1: AHP Application

Analytical Hierarchy Process (AHP) [10] which is a well-known multi criteria decision making technique is selected as the ranking method. When the manual approximation method is used [11], AHP can be applied in six steps at minimum. The number of steps increases as the level of AHP model increases which is actually determined in the first step: Hierarchial problem model construction. The minimum number of levels in an AHP hierarchy is three in that the first level represents the overall goal, the second level represents the objectives or the categories and the last level represents the alternatives.

After constructing the problem model, the second step involves constructing a pairwise comparison matrix for the second level objectives or categories and the third step involves constructing a pairwise comparison matrix for the third level alternatives. Then, the comparison matrix of the alternatives is normalized in the fourth step according to a specific procedure which will be detailed later in this chapter. The fifth step involves the application of a consistency check algorithm to detect inconsistent judgments which can then be corrected to increase the accuracy of the method. Finally, the sixth step utilizes the normalized matrix produced in step four and the comparison matrix in step two to obtain the overall priority ranking among the alternatives.

It should be pointed out that in the proposed methodology, we combined the application of step there and four above into a single step but added an additional step because of the introduction of one more level to the minimal three level hierarchy. Thus, the number of steps remains at six. The explanation of these steps will be given in the succeeding sections.

The reasons for selection of AHP method are detailed below.

Selection of the ranking method: Software selection is a problem that requires a multi-criteria decision making (MCDM) method to be applied. The most frequently used methods in the software selection field are the AHP and the WAS (Weighted Average Sum) methods. Also, fuzzy based approaches have been tried in this area [8].

These three alternatives are considered in this study and AHP is selected as the most proper method because of the following points:

Strengths of AHP concerning the proposed methodology:

- AHP can be applied in a decision making situation where qualitative factors are dominant in number over quantitative factors.

- AHP organizes criteria in a hierarchy which can easily be mapped to categorized software characteristics.

- AHP procedures are suitable for group decision making and help discussions to center on objectives rather than on alternatives [85].

- Pairwise comparisons used in AHP are more accurate and more defensible then arbitrarily assigning a numerical priority to a criterion [11].

- Expressing the comparison using words according to the Saaty's verbal scale is more appropriate to support the inexact judgments of relative CASE tool priorities.

- AHP includes built-in consistency tests [84].

- AHP allows discussion to continue until all available and pertinent information has been considered [85].

Also several weaknesses of the method exist according to Jadhav and Sonar [8]:

Weaknesses of AHP:

- AHP is time consuming because of the high number of pairwise comparisons which increase as the number of alternatives increase. (N(N-1)/2 pairwise comparisons for N alternatives)

- When the number of alternatives or criteria changes, priorities of alternatives should be reevaluated.

- Addition or deletion of alternatives can lead to changes in the final rank [4].

Because of the usage of progressive and rigorous screening in the methodology, the alternatives that can pass to the ranking phase is minimized which refutes the effect of the first weakness. However, the risks involved with the second and third weaknesses are valid but the advantages of the method overwhelm these points.

The WAS method: The weighted average sum method simply involves giving a score and weight for each alternative to find the best one by multiplying these. It is very easy and fast to use however it has serious limitations two of which are considered sufficient for its disqualification. These are given below:

- Since the output of WAS is real numbers, the investigators may wrongly assume that a result of a measured capability can indicate differences in ratios.
  For instance, if we assume that we have found the score of a weighted criterion of a CASE tool as 2, this does not mean that it is exactly twice as better than a CASE tool which has a score of 1 for the same criterion. Therefore, we can conclude that the scores which the method gives do not induce relative superiority differences between the tools [1].

- When the number of criteria is high, it is very difficult for the evaluators to assign weights for each of them because of the limitations of human mind when dealing with multiple alternatives [1].

Fuzzy based approaches: Fuzzy based approaches for decision making in the software selection area were used in relatively few studies however they were found successful and promising. This is attributed to the method's ability to accommodate the vagueness and ambiguity that are inherent in human decision making.

However, the method requires computation of fuzzy appropriateness index and ranking values for all alternatives which is a very tedious work [8]. In case of existence of a dedicated tool for fuzzy calculations, this method may also be used but such a tool couldn't be found by the author.

Considering these findings, a four level AHP implementation was chosen as the most suitable ranking method. It is also important to note that making the rankings according to pure human judgment in a discussion meeting was also an option. However, when the possible number of alternatives and limits of the human decision making capabilities are considered, this approach was found to be much more error-prone than applying a multiple criteria decision model like AHP.

AHP is suggested to be conducted according to the manual approximation method detailed in the article by Jayaswal, Patton and Forman [11] or the real eigenvector method which is the original implementation by Saaty and gives the most accurate results.

However, using the eigenvector method requires extensive calculations and only feasible with the help of an AHP tool like the "Expert Choice" [12]. Expert Choice is a well known tool in the area and has the additional benefit of showing the consistency values during the comparisons.

Shaped by the manual approximation method, this step is composed of six sequential sub-steps given below. It is important to note that this is not the exact replication of the Jayaswal's procedure. Some flexibility is removed in accordance with the methodology however the main principles remain the same.

- Hierarchical problem model construction
- Derivation of ratio scale priorities for the categories
- Derivation of ratio scale priorities for the criteria
- Derivation of ratio scale priorities for the alternatives
- Consistency ratio estimation
- Overall priority ranking

3.7.1.1 Hierarchical Problem Model Construction

When starting to solve a multi-criteria decision making problem using AHP, the first step is constructing the problem hierarchy. The hierarchy is usually depicted in a diagram where the overall goal is placed on top and the alternatives in each layer are connected to their parents finally connecting to the goal. In the figure below, the proposed hierarchy within this methodology is given in the case of nine low level criteria that are arbitrarily numbered and three tool alternatives. (Figure 15)

This is a four level hierarchy with levels: The Goal, Categories, Criteria and Alternatives. The goal represents the organization's aim to apply the ranking that is selecting the best CASE tool.

The categories in level 2 represent the criteria types proposed with the methodology which may be extended by the user. If there is such an extension, then the categories in the hierarchy should be changed to reflect the new type set. In level 3, low level criteria are presented which are connected only to their type. In some AHP implementations elements at this level may be connected to all of the nodes above them however this is not mandatory. In level 4, alternatives that is the tool candidates are presented which are connected to all of the criteria above them.

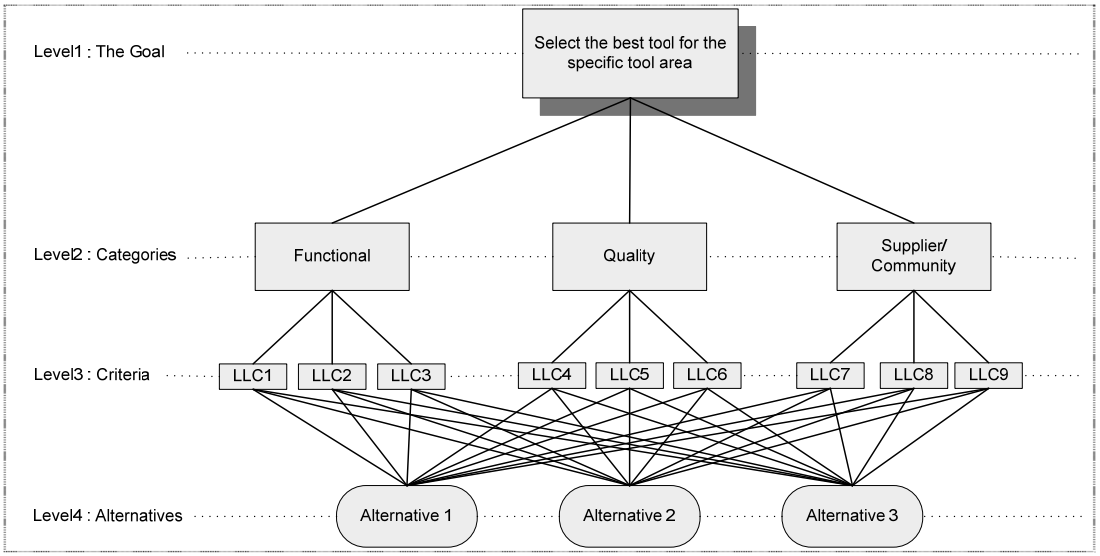

Figure 15: Proposed AHP problem hierarchy

After constructing this hierarchy, the organization can pass to the next step.

3.7.1.2 Derivation of Ratio Scale Priorities for the Categories

In this step, relative importance of the categories should be judged by the evaluators which is accomplished by utilizing the "Fundamental Verbal Scale for Pairwise Comparison" given in Table 5. The evaluators should make the below comparisons one by one and insert the corresponding values from the verbal scale into the category comparison table.

Comparisons to be made:

- Functional versus Quality
- Functional versus Supplier/Community
- Quality versus Supplier/Community

Then the row-wise totals are computed and weighted for each row and inserted into the category comparison table. The format of this table is given below. (Table 8) It is suggested to use three decimal places for all of the calculations in this activity.

Table 8: Category comparison table format

Comparison	**Functional**	**Quality**	**Supplier/ Community**	**Approximate Weight**
Functional	1	Result of verbal scale comparison	Result of verbal scale comparison	Row total 1/ Total
Quality	Result of verbal scale comparison	1	Result of verbal scale comparison	Row total 2/ Total
Supplier/ Community	Result of verbal scale comparison	Result of verbal scale comparison	1	Row total 3/ Total
Total				Total = Row total 1 + 2 + 3

Approximate weight of each row represents the quantified relative importance of each category.

3.7.1.3 Derivation of Ratio Scale Priorities for the Criteria

In this step, relative priorities of the criteria in regard to their parent category are determined. The same method used for the categories is used for criteria comparisons. That is a comparison table is formed using the verbal scale. The following comparisons should be made in case of three criteria for each category as in Figure 15.

Comparisons to be made:

- LLC1 versus LLC2 (for functional category)
- LLC1 versus LLC3 (for functional category)
- LLC2 versus LLC3 (for functional category)
- LLC4 versus LLC5 (for quality category)
- LLC4 versus LLC6 (for quality category)
- LLC5 versus LLC6 (for quality category)
- LLC7 versus LLC8 (for supplier/community category)
- LLC7 versus LLC9 (for supplier/community category)
- LLC8 versus LLC9 (for supplier/community category)

Three tables should be constructed corresponding to each category. The format of the tables will be the same as Table 8 except the row and columns will be filled by the criteria IDs. It is important to note that the total of criterion weights should be one for each category. This means that if we add the weights of every criterion, we would get three. After determining the weights of the criteria, the evaluation team may pass to the next step.

3.7.1.4 Derivation of Ratio Scale Priorities for the Alternatives

Calculation of the priorities for the alternatives is different than the scheme used in the previous steps. The original method of Saaty involves mathematical eigenvalue and eigenvector calculations for this phase [10]. However, if the evaluator team lacks suitable software for this purpose, the following approximation method is proposed in accordance with Jayaswal [11]. This procedure is fast and provides very close results to the eigenvalue computation if the consistency of comparisons is high which is calculated in the next step.

Procedure for priority approximation:

1. Collect the comparison data determined in "Evaluation and Comparison" activity previously for the first criterion.

2. Construct the synthesis table given below (Table 9) for the criterion.

3. Sum the verbal scale values column-wise to get the totals for each column.

4. Divide each cell by its column total and record the result in the corresponding synthesis cell. (The synthesis section of the table is called the "normalized pairwise comparison matrix" in AHP)

5. Find the average of the elements in each row of the synthesis section. These averages represent tool superiority in regard to the specific criterion being considered.

6. Repeat the procedure for other criteria.

Table 9 shows the format of the synthesis table proposed.

Table 9: Synthesis table format

Item Name	**Description**		
Criterion ID	The number of the criterion		
Criterion Name	The name of the criterion		
Comparison (this section comes from the evaluation activity)	**Candidate 1**	**Candidate 2**	**Candidate 3**
Candidate 1	1 (A)	Result of verbal scale comparison (D)	Result of verbal scale comparison (H)
Candidate 2	Result of verbal scale comparison (B)	1 (E)	Result of verbal scale comparison (I)
Candidate 3	Result of verbal scale comparison (C)	Result of verbal scale comparison (F)	1 (J)
Total	Column total 1 (CT1)	Column total 2 (CT2)	Column total 3 (CT3)

Synthesis	**Candidate 1**	**Candidate 2**	**Candidate 3**	**Row Average**
Candidate 1	A/CT1	D/CT2	H/CT3	Av1
Candidate 2	B/CT1	E/CT2	I/CT3	Av2
Candidate 3	C/CT1	F/CT2	J/CT3	Av3
Total				1.000

An example from the case study is given below for the synthesis table which should be prepared for every low level criterion. (Figure 16)

Criterion ID	LLC16				
Criterion Name	Role-based user management				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	1/5	1/7	1/8	
Hudson	5	1	1/2	1/3	
QuickBuild	7	2	1	1/2	
TeamCity	8	3	2	1	
Total	21.000	6.200	3.643	1.958	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.048	0.032	0.039	0.064	0.046
Hudson	0.238	0.161	0.137	0.170	0.177
QuickBuild	0.333	0.323	0.274	0.255	0.296
TeamCity	0.381	0.484	0.549	0.511	0.481
Total					1.000

Figure 16: Example from the case study

Afterwards, the team may pass to the next step where the consistencies of the previous comparisons will be checked.

3.7.1.5 Consistency Ratio Estimation

The AHP method tolerates inconsistency in the comparisons. However, to be as accurate as possible, it is suggested to keep the inconsistency low. AHP offers a procedure to compute the index of consistency for each comparison made according to the verbal scale.

In this step, consistency index calculations for all of the pairwise comparisons are performed to find and correct the inconsistent judgments. The following procedure should be followed to compute the index and ratio values. A consistency ratio less than or equal to 0.1 is considered reasonable.

Procedure for consistency estimation:

1. For the first criterion, construct the consistency check table from the template given below. (Table 10)

2. Multiply each value in a column in the comparison section of the synthesis table with the corresponding candidate priority given as the row average and insert into the consistency check table according to the template.

3. Sum each row of the consistency check table and record in the corresponding cell.

4. Divide each total with the corresponding priority value of the row alternative and insert the values found into the "Division" column.

5. Compute the average of the division values. (DAv: Division Average)

6. Compute the consistency index (CI) according to the formula below.

   CI = (DAv-n) / (n-1)

   where n = the number of items being compared

7. Compute the consistency ratio (CR) according to the formula below.

   CR = CI / RI

   where Random Index (RI) is the average random consistency index given by Saaty which depends on the number of elements being compared according to the table below.

Table 10: RI values [10]

n	3	4	5	6	7	8
RI	0.58	0.90	1.12	1.24	1.32	1.41

As previously stated, CR ≤ 0.1 is considered acceptable.

8. Repeat the procedure for other criteria.

Table 11 shows the format of the consistency check table proposed.

Table 11: Consistency check table format

Item Name	**Description**				
Criterion ID	The number of the criterion				
Criterion Name	The name of the criterion				
Consistency	**Candidate 1**	**Candidate 2**	**Candidate 3**	**Totals**	**Division**
Candidate 1	A x Av1	D x Av2	H x Av3	Rowtotal1	Rowtotal1 /Av1
Candidate 2	B x Av1	E x Av2	I x Av3	Rowtotal2	Rowtotal2 /Av2
Candidate 3	C x Av1	F x Av2	J x Av3	Rowtotal3	Rowtotal3 /Av3
Average					DAv
CI	(DAv-n) / (n-1) = (DAv – 3) / 2 (n=3 for this example)				
CR	CI / RI = CI / 0.58 (According to Table 10)				

An example for the consistency check table is given below. (Figure 17)

Criterion ID	LLC2					
Criterion Name	Extension mechanisms					
Consistency	CruiseControl	Hudson	QuickBuild	TeamCity	Totals	Division
CruiseControl	1 x 0.501	3 x 0.219	5 x 0.062	3 x 0.219	2.249	4.489
Hudson	1/3 x 0.501	1 x 0.219	5 x 0.062	1 x 0.219	0.915	4.178
QuickBuild	1/5 x 0.501	1/5 x 0.219	1 x 0.062	1/5 x 0.219	0.221	3.567
TeamCity	1/3 x 0.501	1 x 0.219	5 x 0.062	1 x 0.219	0.915	4.178
Average						4.103
CI	(4.103 – 4) / 3 = 0.034					
CR	0.034 / 0.90 = 0.038					

Figure 17: Example from the case study

If the consistency ratio of a comparison is found to be high, then the judgments made in the comparison should be reevaluated.

3.7.1.6 Overall Priority Ranking

The relative priority of all the alternatives in regard to criteria is computed before entering this step. Moreover, the relative priority of the criteria against each other and the relative priority of the categories are also ready. In this last step of the methodology, the evaluators should combine these findings in order to obtain the final ranking of the tools.

First, the ranking of the tools in regard to the categories should be found. For this purpose, the relative priorities of the criteria belonging to a category should be multiplied with the relative priorities of the alternatives. If we consider the example given in Figure 15, the rankings for the functional category which contains LLC1, LLC2 and LLC3 can be computed with the equations below.

Weight of Alternative 1 for the functional category = (Weight of LLC1 x Weight of Alternative 1 in regard to LLC1) + (Weight of LLC2 x Weight of Alternative 1 in regard to LLC2) + (Weight of LLC3 x Weight of Alternative 1 in regard to LLC3)

Weight of Alternative 2 for the functional category = (Weight of LLC1 x Weight of Alternative 2 in regard to LLC1) + (Weight of LLC2 x Weight of Alternative 2 in regard to LLC2) + (Weight of LLC3 x Weight of Alternative 2 in regard to LLC3)

Weight of Alternative 3 for the functional category = (Weight of LLC1 x Weight of Alternative 3 in regard to LLC1) + (Weight of LLC2 x Weight of Alternative 3 in regard to LLC2) + (Weight of LLC3 x Weight of Alternative 3 in regard to LLC3)

After making these computations, we can rank the alternatives in the functional category. Ranking in the other categories is likewise and straightforward.

Then, the overall ranking of the tools can easily be found by multiplying the tool weights in regard to categories and category weights. For instance, the equation below can be used to compute the overall weight of Alternative 1 in our example model.

Overall weight of Alternative 1 = (Weight of the functional category x Weight of Alternative 1 for the functional category) + (Weight of the quality category x Weight of Alternative 1 for the quality category) + (Weight of the supplier/community category x Weight of Alternative 1 for the supplier/community category)

The weights of the other alternatives can be computed likewise and overall rankings can be presented.

3.7.2 Step2: Final Selection

In the last step, final assessment about the tools should be made according to the ranking devised in the previous step. This step is necessary because of the following reasons:

- The final relative priorities of the tools may be very close to each other or may even be the same. Even if the AHP method provides reliable results with human comparisons as input, those comparisons remain subjective so close results do not impose absolute superiority of a tool over another.

- The proposed methodology provides a benefit oriented comparison between the tools. The tool cost was only considered as a high level criterion at the beginning to screen the tools that have considerably exceeding prices. However, the combined tool prices in case of a multiple tool selection or the other costs involved like the training and maintenance costs were not taken into account. In this step, the so called tool costs can be reevaluated by the evaluators and cheaper tools may be selected from nearly weighted alternatives although they may not be the first.

- The tools' interoperability and compatibility requirements with the environment they will work in were considered as potential candidates for high level criteria. However, in case of a tool combination selection, the integration requirement of the tools between each other was not taken into account since this was impossible to evaluate before the selection of the tools. In this step, the integration possibilities between the tools can be examined and a tool combination that provides the most powerful and seamless integration can be selected even if this combination is not formed from the tools that are ranked as first in each tool area.

- If the objective of the organization applying this methodology was to decide whether they will keep a tool or replace it, then this decision can be made at this step. The evaluators should check the ranking, overall priority weight and the integration ability of their tool to the other tools that are selected (in case of a multiple tool selection) and make a judgment about replacement. When the possible expenditure that will be spent on the new tool and possible time that will be dedicated to learn it are considered, it is suggested that the existing tool be kept if it is not too behind its competitors in the ranking and it has integrations for the other tools that will be acquired.

So this step is the last chance to make a final evaluation considering the tool weightings, cost, interoperability and other issues not mentioned in the methodology. This decision should be made by ensuring an evaluator consensus.

After making the decision, the tools may be studied in a pilot project for validation purposes before making their acquisition. This study may reveal incorrect criteria evaluations such as tool's inability to satisfy a criterion although it has gained a check in the screening phase for that criterion. Also, some points that are missed from consideration when defining the criteria may be discovered. However, this work is only suggested if a suitable project and necessary time and resources are available. The influence of human factors and other complications concerning pilot projects are discussed in detail by Kitchenham and Kunda [89, 95].

After this step, the organization is considered to have finished the application of the methodology. An informal "lessons learned" document may optionally be prepared at this step to guide future evaluations. The case study which demonstrates all the steps of the methodology on a real organization is given in the next chapter.

CHAPTER 4

APPLICATION OF THE METHODOLOGY ON A CASE STUDY

The organization that is the subject of this case study is a real institution participating in several sectors. The IT department of the institution is actively developing, supporting and maintaining software that is being used externally from web and internally by other departments in the institution. Seventeen active developers are working in the department. One of them is also responsible for the deployment of developed systems on production systems and will be called as the deployer throughout this chapter. Mainly Java and J2EE technologies are being used. The IT department was planning to engage CASE software in its continuous integration process which is a practice for which there are many CASE tools claiming to support complete or partial aspects. The decision makers in the department had to select among the alternatives which will not need extensive customization, address all the needs and be cost effective. Therefore the department was a reasonable candidate for application of the proposed methodology.

The research questions of the CASE study were:

- Is the proposed methodology feasible to use in the context of the subject of the case study? (Are all of the activities applicable and is the methodology able to produce a result?)
- Can the methodology be finished in the expected time frame?
- How is the feedback of the evaluators and stakeholders about the methodology?
- Does the methodology enhance the evaluators' knowledge of the organizational requirements and CASE technology as proposed?

- How is the result of the methodology compared to the result of an ad hoc decision in case of not using a methodology?
- How is the confidence of the evaluators and management about the results of the methodology? (Will the tools found be actually used?)

These questions are answered in the discussion part at the end of this chapter.

The author of this thesis is also working in this institution as a software specialist and was the one who offered the acquisition of CASE tools as a solution for the institution's problems and using the proposed methodology for selecting the best set. The details of the proposed methodology were explained to the department managers who then gave permission for the application and authorized the author of this thesis for conduction. Afterwards, an evaluation team was formed the members of which will be detailed in the succeeding sections. As stated, the overall purpose was the determination of a CASE tool solution which will solve the integration problems being faced.

In the rest of this thesis, the details of the application of the proposed methodology will be presented. The organization's name will not be disclosed; instead "the department" will be used to refer to the IT department of the organization. Also, the workers' roles will be referred when needed instead of their real names.

4.1 ACTIVITY 1: INITIATION

4.1.1 Step 1: Rationale Determination

The rationale of the department fits to the first and second scenario given in the methodology that is “the organization may want to automate a complete software development phase or a part of it” and “the organization may want to replace a CASE tool with a better alternative”.

Continuous integration was already being conducted in the department but by manual means. There was some automation in the subpractices that form continuous integration however they were not integrated as a whole.

If we divide continuous integration into three necessary distinct areas (as explained in the literature survey section) as continuous integration server area, version control and build area, we can further analyze the problems being faced. The continuous integration server tool area will be abbreviated as CI server tool area in the rest of the thesis. First of all, the department did not have a CI server so automatic starting of builds was not being accomplished. The deployer was collecting the work of everyone, integrating them and making a deployment if everything was successful. This integration activity was not being performed on regular intervals. It was being done when it was required to add functionality or solve a problem in the system.

During the integration, problems with the code were appearing occasionally and solving these problems was causing unexpected delays if they were hard to find. This was causing severe problems if the change requested to the system was immediate.

Also, the team was having problems with their version control system, VSS (Visual Source Safe). The main problem of the system was that it was not very successful at handling concurrent development however this was not concerning continuous integration. One drawback of VSS affecting continuous integration was the lack of system notification that warns the users to commit their files after they finish working on them. Some systems warn the user to commit when they exit their workbenches. A similar functionality was needed because developers were sometimes forgetting to commit their files into the repository even if they finished working on them. Thus the changes in those files were not making their way into the integration build which was causing nonexistence of expected functionality in the system. Finding the source of this nonexistence was also causing loss of valuable time.

Developers were using the open source free tool named Ant as their build tool. They were using it to package their applications on their workbenches. The department did not have a problem with this tool and the packaging functionality it had. However, to see the new possibilities in the technology, the build area was also requested to be evaluated.

According to the proposed methodology, it is required to form a problem statement at the end of this activity. The situation of the department was summarized in the following problem statement by the evaluators.

Problem statement:

The department is having difficulty and unnecessary delays in their integration process. This is due to the human controlled (ad hoc) start and conduction of the process. The integration of the changes made by individual developers to a central system is being performed by a developer having the deployer role. The work collected from each developer may not include the final work or the integration of the components may cause compilation failures which can only be caught when the deployer is making the integration build.

Complexities involved with finding the source of the problem makes the situation worse and further delays the presentation of the added functionality to the system which is sometimes requested immediately.

Automation of the integration process as a whole by using CASE technology is expected to be a solution for the problem. Writing a tool from scratch instead of investing on CASE software is not an option since the department does not have the necessary experience or time for such an effort.

4.1.2 Step 2: Commitment Determination

In this step, the department's commitment for the adoption of a CASE tool was investigated according to the proposed methodology. The answers to the question set given by the methodology which were answered by the evaluators are given below.

Answers to the Commitment Requirements Question Set:

1. Has the management fully agreed on acquisition and implementation of a CASE tool?

   The management is also aware of the problems being faced in the integration process. The developers are usually blaming the integration failures for the delays in providing the requested functionalities. The managers expect an increase in the development efficiency by adoption of CASE tool(s) in the integration area.

2. Has the management fully agreed on funding a CASE tool?

   After mutual acceptance of the potential benefits of the CASE tool technology to be used, management agreed on funding the acquisition. A maximum and preferred amount of expenditure from the organization's annual budget is determined.

3. Has the management fully agreed on supporting the selection process?

   When it is explained to the management that there are too many alternatives in the market and ad hoc processes may cause selection of a wrong tool which means wasted expenditure, the management was voluntary for the application of the proposed methodology.

4. Has the development team agreed on acquisition and implementation of a CASE tool?

   Since the build failures were causing extra "find and fix" work for the team, they were eager for the adoption however some of them had negative preconceptions for the CASE technology because of their experiences.

After obtaining these results for the commitment question set, it was concluded that the department is at a sufficient commitment level for continuation to the methodology.

4.1.3 Step 3: Methodological Constraints Determination

In this step, the department's constraints concerning the application of the methodology were determined. The question set given by the proposed methodology was answered.

Answers to the Methodological Constraints Question Set:

1. What is the number of personnel that can be delegated as the evaluators?

   The department decided that an evaluator team which is composed of three employees will be enough. One of them will be from the management.

2. How much time can the evaluators allocate for the methodology work?

   The management stated that the evaluators should spend at most 3 hours a day for the methodology work. Urgent issues about development have precedence over the application of the methodology.

3. How much time can the organization devote to the selection process totally?

   The department is not in a hurry for the application of the methodology since there is no deadline waiting for a project milestone. However, the problems in the integration process are causing continuous trouble for the development team and acquisitions are usually performed in the first two months of the year in the organization. Therefore, 1.5 (one and a half) month is determined as available to apply the methodology from start to finish.

4.1.4 Step 4: Evaluation Team Formation

The department has created the evaluation team which included three employees due to management decision.

One of the evaluators was the author of this thesis who also was the coordinator of application. One of them was the assistant manager of the department and the last was a senior developer who only made decision making contribution. It was stated that this team can be treated as representative for the entire department.

4.2 ACTIVITY 2: THE HIGH LEVEL EVALUATION CRITERIA DEFINITION

4.2.1 Step 1: Constraints Determination

In this step, the department determined its constraints about the acquisition process according to the methodology. The constraints had to resemble the "must" characteristics of the system and had to be expressed in three groups as: organizational, environmental and development phase related constraints. Here are the answers that evaluators gave to the questions in the question set provided by the methodology.

4.2.1.1 Answers to the organizational constraints question set:

1. *What is the maximum expenditure we can spend on the tool?*

   The maximum amount of expenditure that the department can spend on the tool(s) is determined as 40000$. However, the preferred cost is about 32000$.

2. *What is the number of licenses needed?*

   Since there are seventeen active developers (one of them is the deployer), roughly nine floating or seventeen node-locked licenses will be required. Since everyone will be involved in the continuous integration process, the authorized license requirement will also be seventeen. These numbers will be used to estimate tool costs where required.

3. *How much time should be required to fully integrate the tool into the organization's development practices?*

   It has been decided that it should take at most two months to fully implement and integrate the tool into the department's processes.

4. *Should the tool conform to an international standard that is adopted in the organization?*

   The department's processes should conform to the standards ISO 9001, ISO 27001 and ISO 20000. These are the standards that the department is continuously verified by the authorities. The tools to be acquired should possess security, auditability and backup facilities for this constraint to be satisfied.

4.2.1.2 Answers to the environmental constraints question set:

1. *Do we need a tool that supports a specific processor architecture?*

   X86 type architectures are being used in the department therefore the intended CASE tool(s) should support X86.

2. *Do we need a tool that supports a specific CPU bus size?*

   Clients are utilizing 32 bit processors, servers are utilizing 64 bit processors so the CASE tool(s) that would be installed on the servers should be compatible with 64 bit systems and the CASE tool(s) that will be installed on the clients should be compatible with 32 bit systems.

3. *Do we have a constraint on the CPU usage of the tool?*

   The department does not have a constraint on the CPU usage of the tool.

4. *Do we have a constraint on the disk usage of the tool?*

   The department does not have a constraint on the disk usage of the tool.

5. *Do we have a constraint on the memory usage of the tool?*

   The departmental constraint about the CASE tool(s) is that an individual tool's memory consumption should not go beyond 1 GB.

6. *Do we have a constraint on the network usage of the tool?*

   The department has a quite capable network so does not have a constraint about network usage of the tool.

7. *Do we need a tool that supports a specific operating system for the server?*

   The tool(s) that would be installed on the servers should support Windows Server 2003 operating system.

8. *Do we need a tool that supports a specific operating system for the clients?*

   The tool(s) that would be installed on the clients should support Windows XP SP3 operating system.

9. *Do we need a tool that should integrate with a specific software being used in the organization?*

   No mandatory integration is required other than the operating system integration which is covered above. However, Eclipse integration is desirable which may be included as a low level criterion later. In the case of a tool combination selection, the integration possibilities between the tools themselves will be covered at the final selection step at the end.

10. *Do we need a tool that should be compatible with a specific software being used in the organization?*

    The applicable tools should be compatible with Java platforms, the application server being used in the department (Oracle Weblogic 9.2) and also with Internet Explorer 7, Firefox 3.0 for the tool(s) that have a web interface.

4.2.1.3 Answers to the Development Phase Related Constraints Question Set:

1. *In which software development phase or practice will the CASE tool be used?*

   The CASE tool(s) will be used for the continuous integration practice which will be integrated into the current software development practices in the department.

2. *To which subpractices can the practice be divided?*

   After the examination, the department concluded that the practice of continuous integration can be further subdivided into three fundamental areas: the CI server area, the version control area and the build area. This set satisfies the main goal which is automatic invocation of a build cycle upon a change in the system that can generate feedback in return.

   The continuous integration systems may also include automated testing, inspection and deployment facilities however these are not mandatory and can be added to the system afterwards.

4.2.2 Step 2: Existing Toolset Examination

This step was given as optional in the methodology since a CASE tool might not exist in the organization. However, in the department two CASE tools were being used corresponding to two subpractices of the continuous integration process namely to the version control and build areas. Therefore this step was essential for the department since these might be replaced. Following answers were prepared for the question set given in the methodology.

4.2.2.1 Answers to the Existing Toolset Examination Question Set:

1. *Which tools are currently being used in the organization for the intended field?*

   When we divide the continuous integration into three as the CI server area, version control area and build area, we see that the department is already using CASE software for the version control and build operations. The version control tool being used is VSS (Visual Source Safe) and the build tool being used is Ant.

VSS is a commercial tool from Microsoft and Ant is an open source tool that can be used by conforming to the Apache license.

2. *For each of these tools, what are the functionalities or good quality characteristics that the users like and benefit from?*

   Given in the "Positive Qualities of Existing Tools" section below.

3. *For each of these tools, what are the drawbacks?*

   Given in the "Drawbacks of Existing Tools" section below.

4. *For each of these tools, what are the desired additional functionalities?*

   Given in the "Desired Additional Functionality" section below.

4.2.2.2 Positive Qualities of Existing Tools

- For the version control tool VSS
  - Easy to install
  - Easy to configure
  - Easy to use
- For the build tool Ant
  - Integrated with the Eclipse platform
  - Easy to configure due to its XML base configuration files

4.2.2.3 Drawbacks of Existing Tools

- For the version control tool VSS

    - Risky usage model (in case of a failure during commit, the database may be left in a corrupted state)
    - No directory versioning
    - No change set support
    - Lack of command line functionality (this is needed for writing batch scripts for routine activities)

- For the build tool Ant

    - The XML based build files can get quite large during the project which makes them difficult to manage
    - It does not have fault handling rules or persistence of state so it cannot be used for workflow type operations

4.2.2.4 Desired Additional Functionality

- For the version control tool

    - A strong permission model
    - Ability to access the versions without keeping them in local disk
    - Directory versioning
    - Change set support
    - Extensive command line library

- For the build tool

  No additional desired functionality can be extracted however it is desirable that the tool to be acquired does not have the drawbacks mentioned in the previous section for Ant.

This concludes the existing toolset examination step. It can be observed that the department is having problems especially with the version control tool being used but they also want to evaluate the build tool area.

4.2.3 Step 3: Tool Area to Search Determination

In this step of the Evaluation Criteria Definition activity, the evaluators tried to identify the technology areas for which CASE tools are developed in the market. When determining the development phase related constraints, the evaluators have arrived to the conclusion that the continuous integration process may be divided into the three areas as; CI server, version control and build. Then the literature was searched to find whether these activities are individually covered by different tools or a tool covers a combination of them itself. Therefore the possible options were:

1. One tool for each area making three tools totally
2. One tool that has both the CI server functionality and the version control functionality
   One tool for the build functionality
3. One tool that has both the CI server functionality and the build functionality
   One tool for the version control functionality
4. One tool that has both the version control functionality and the build functionality
   One tool for the CI server functionality
5. One tool that covers all the functionalities

After searching information sources (primarily the web), the department couldn't find CASE tools that fit to the options given as 2, 3, 4 and 5 above. So at the end of this step, it is concluded that there are 3 tool areas to consider: CI server area, version control tool area and build tool area. One additional observation was that all of the version control tools can provide the simple functionality required by a continuous integration practice. This includes the VSS that is being used in the organization however the problems being faced in this area directed the evaluators to look for configuration management solutions instead of just version control. Therefore, the version control tool area was replaced with the broader configuration management tool area which was examined from then on.

4.2.4 Step 4: High level Criteria Determination for Each Tool Area

In this step, the department had to group the requirements into the tool areas which were finalized in the previous step. The requirements were gathered from the outputs of the "constraints determination" step and the "existing toolset examination" step. Moreover, they had to be converted to criteria in the form that is explained by the methodology. During this conversion, some of the requirements were matched to multiple criteria and some of them were rephrased. Explanation is given for each of the criteria that are changed from its source requirement in the details below.

4.2.4.1 General High Level Criteria

- HLC1: The CASE tool(s)' purchase and one year maintenance prices shall cost to 40000$ maximum if the organization purchases nine floating or seventeen node locked licenses.

  Note: The cost and license number constraints are combined as a single criterion here.

- HLC2: The CASE tool(s) shall be able to be implemented in two months.

  Note: The constraint that mandates conformance to ISO 9001, 27001 and 20000 is converted to the three criteria given below.

- HLC3: The CASE tool(s) shall include a security or a permission model (this originates from the security requirement mandated by the standards that is; not everyone should be able to use the tool)
  Exception: Not applicable to the build tool

- HLC4: The CASE tool(s) shall include a logging mechanism (this originates from the auditability requirement imposed by the standards that is the usage history of the tool should be reportable)

- HLC5: The CASE tool(s) shall include a backup mechanism (this originates from risk mitigating actions imposed by the standards)
  Exception: Not applicable to the CI server tool

- HLC6: The CASE tool(s) shall support X86 architectures.

- HLC7: The CASE tool(s) shall support 32 bit systems for their client software.

- HLC8: The CASE tool(s) shall support 64 bit systems for their server software.

- HLC9: The CASE tool(s) shall be able to work with a maximum of 1GB memory consumption.

- HLC10: The CASE tool(s) shall support Windows Server 2003 operating system for their server software.
  Exception: The tools that does not have a server component

- HLC11: The CASE tool(s) shall support Windows XP SP3 operating system for their client software.
  Exception: The tools that does not have a client component that is the tools that only work on a single server or multiple servers.

- HLC12: The CASE tool(s) shall support Internet Explorer 7 and Firefox 3.0 for their web interfaces.
  Exception: This is a criterion for the tools that do not have a fat client component and can only be used via their web interface.

- HLC13: The CASE tool(s) shall support Oracle Weblogic application server 9.2.
  Exception: Not applicable to the configuration management tool

- HLC14: The CASE tool(s) shall support JAVA platforms.

4.2.4.2 High Level Criteria for CI Server Area:

- HLC15: The tool shall be developed for functioning as a continuous integration server.

  Note: The department has added this criterion because it is found that several tools exist in the market which entails the CI server capability but also developed to handle a complete collaboration process including many unrelated functionality as release management or build artifact management. The department doesn't want the extra clutter coming from unrelated functionality.

4.2.4.3 High Level Criteria for the Configuration Management Tool Area:

Note: The drawback reported as the risky usage of the current tool is reflected with the below criterion which states that the atomic commit functionality is required. This functionality is found through web by the evaluators and its definition is given under the criterion.

- HLC16: The tool shall support atomic commits.
  Atomic commit: An *atomic commit* is an operation in which a set of distinct changes is applied in a single transaction. If a failure occurs before the transaction finishes, then all the changes pertaining to the commit operation are rolled back leaving the system in a consistent state.

- HLC17: The CASE tool shall support directory versioning.
  Directory versioning: The ability to give succeeding revision numbers to directories and keep their namespace history in order to be able to return to a previous state.

- HLC18: The CASE tool shall support change sets.
  Change set: A logical container of file and directory versions that belong to a specific task.

- HLC19: The CASE tool shall include a command line interface

- HLC20: The CASE tool shall support access to version controlled code remotely. That is the CASE tool shall include a web interface or other means to access code that is not in the local network.

4.2.4.4 High Level Criteria for the Build Tool Area:

- HLC21: The CASE tool shall be developed for JAVA building and packaging.
  Note: As in HLC15, this criterion is created for screening the tools that contain unnecessary functionalities.

So, after this step the department was considered to have finished defining its high level evaluation criteria for tool selection and evaluators might pass to the prescreening phase.

It should also be pointed out that these criteria were validated by the assistant manager of the department who also was a member of the evaluation team.

4.3 ACTIVITY 3: PRESCREENING

After the evaluators had finished forming the high level criteria set for each tool area, the prescreening activity was started. In this activity, the department had to find the CASE tools developed for each tool area in the market. Then the evaluators had to use the high level criteria to eliminate the ones that do not meet the department's needs.

4.3.1 Step1: Tool Information Gathering for Each Tool Area

The tools below were found in the market for each tool area. The internet was used as the information source.

1. **CI Server Tool Area:** The table below (Table 12) includes tool information for the thirty tools found in the market.

Table 12: CI Server tool information table

Tool Name	Producer-Vendor	License	Tool Site	Version
CruiseControl	Originally developed by ThoughtWorks	BSD-style license	[25]	2.8.2
Hudson	Originally developed by a worker in Sun	MIT license	[26]	1.341
Continuum	Apache Software Foundation	Apache 2.0 license	[27]	1.2.3
Luntbuild	Javaforge	Public Domain	[28]	1.6.3
QuickBuild	PmEase	Commercial	[29]	2.0.15
Cruise	ThoughtWorks	Commercial	[30]	1.3.2
BuildForge	IBM	Commercial	[31]	7.1
AnthillPro	Urbancode	Commercial	[32]	3.7.1
Gump	Apache Software Foundation	Apache License 2.0	[33]	3.7
Automated Build Studio	AutomatedQA	Commercial	[34]	5.0
Bamboo	Atlassian Software Systems	Commercial	[35]	2.5
Beebox	TechSolCom IT Group	Free	[36]	3.0.3
CABIE	Collabnet	GNU General Public License	[37]	2.0
Cerberus	Anatol Pomozov	Free	[38]	0.7
CruiseControl. NET	Originally produced by ThoughtWorks	BSD-style license	[39]	1.5.0
CruiseControl. rb	Originally produced by ThoughtWorks	Apache License 2.0	[40]	1.4.0
ControlTier	ControlTier development team	Apache License 2.0	[41]	3.4.9
Draco.NET	Draco.NET development team	BSD-style license	[42]	1.5
EasyCIS	Vaclav Zahradnik	Commercial	[43]	1.0.44.70
Electric Commander	Electric Cloud	Commercial	[44]	2.2
FinalBuilder	Vsoft Technologies	Commercial	[45]	6.0
InstallAce	DigiAce	Commercial	[46]	1.0
OpenMake Meister	OpenMake software	Commercial	[47]	7.0
OpenMake Mojo	OpenMake software	Commercial	[48]	7.0
Parabuild	Viewtier Systems	Commercial	[49]	4.0
Pulse	Zutubi	Commercial	[50]	2.0.49
TeamCity	JetBrains	Commercial	[51]	5.0
TFS	Microsoft	Commercial	[52]	2008
Tinderbox	Mozilla Corporation	Mozilla Public License	[53]	2.0

2. **Configuration Management Tool Area:** The table below (Table 13) includes tool information for the twenty two tools found in the market.

Table 13: Configuration management tool information table

Tool Name	Producer-Vendor	License	Tool Site	Version
AccuRev	AccuRev Inc.	Commercial	[54]	4.7.3
BitKeeper	BitMover Inc.	Commercial	[55]	3.2.4
ClearCase	IBM Rational	Commercial	[56]	7.1.1
Synergy	Telelogic (IBM)	Commercial	[57]	7.1
Co-Op	Reliable Software	Commercial	[58]	5.1
Perforce	Perforce Software Inc.	Commercial	[59]	2009
PureCM	PureCM Ltd.	Commercial	[60]	2009-2
Source Anywhere	Dynamsoft Corporation	Commercial	[61]	2.3
Surround SCM	Seapine Software	Commercial	[62]	2010.0.1
Team Foundation Server	Microsoft	Commercial	[63]	2008
Vault	SourceGear LLC	Commercial	[64]	5.0.2
VSS	Microsoft	Commercial	[65]	2005
CVS	The CVS Team	GNU General Public	[66]	1.11.22
Aegis	Peter Miller	GNU GPL	[67]	4.24
Bazaar	Canonical Ltd.	GNU General Public	[68]	2.1.0b4
Darcs	David Roundy	GNU General Public	[69]	1.0.4
Mercurial	Matt Mackall	GNU General Public	[70]	1.4.2
Monotone	Nathaniel Smith, Graydon Hoare	GNU General Public	[71]	0.45
OpenCM	Eros project team	GNU General Public	[72]	0.1.2
Subversion	CollabNet, Inc.	Apache/BSD style	[73]	1.5.6
Svk	Best Practical	Artistic/GPL	[74]	2.2.1
Vesta	Originally developed by Compaq/Digital Systems Research Center	GNU General Public	[75]	2.1.12

3. **Build Tool Area:** The table below (Table 14) includes tool information for the seven tools found in the market.

Table 14: Build tool information table

Tool Name	Producer-Vendor	License	Tool Site	Version
Ant	Apache Software Foundation	Apache License 2.0	[76]	1.8.0RC1
NAnt	NAnt development team	GPL	[77]	0.86
Maven	Apache Software Foundation	Apache License 2.0	[78]	2.2.1
Phing	Phing development team	GNU Lesser Public General License (LGPL)	[79]	2.4.0RC3
Rake	Rake development team	MIT/X Consortium License	[80]	0.8.7
Xcode	Apple Inc.	Commercial	[81]	2.3
Raven	Raven development team	Apache Software License	[82]	1.2

4.3.2 Step 2: Criteria Matching for Each Tool Area

In this step, the author has searched information about the candidate tools to assess whether they satisfy the stated high level criteria or not. This step was actually conducted in parallel to the previous step as stated in the methodology. That is, when the author was searching tool information for the previous table, he also found information regarding the criteria support of the tools and immediately checked the corresponding places in the table below. (Table 15) However, only a few cells of Table 15 were filled this way. All the remaining cells required extensive search by the author which made this step the most time consuming task of the overall methodology even if the columns were ordered according to the order of importance of the criteria.

1. CI Server Tool Area:

Table 15: CI server tool – criterion matching table (checklist)

	HLC ID												
Tool Name	**15**	**14**	**13**	**6**	**10**	**8**	**1**	**12**	**2**	**3**	**4**	**9**	**7**
CruiseControl	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Hudson	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Continuum	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Luntbuild	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QuickBuild	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cruise	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
BuildForge	✓	✓	✓	✓	✓	✓	**X**	ne	ne	ne	ne	ne	ne
AnthillPro	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Gump	✓	✓	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne	ne
Automated Build Studio	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Bamboo	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Beebox	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CABIE	✓	✓	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne	ne
Cerberus	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
CruiseControl .NET	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
CruiseControl .rb	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
ControlTier	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Draco.NET	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
EasyCIS	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Electric Commander	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne	ne	ne	ne
FinalBuilder	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
InstallAce	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Jhbuild	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
OpenMake Meister	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
OpenMake Mojo	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Parabuild	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Pulse	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TeamCity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Team Foundation Server	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Tinderbox	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne

✓: Criterion is successfully satisfied by the tool

X: Criterion is not satisfied by the tool

ne: Not Evaluated

The notes below were written for future reference. They include rejection reasons for the failed tools and some important information for the other tools.

Notes:

- CruiseControl can handle HLC3 (security criterion) by the userid parameter given to the JMX HTTP Adapter [14].
- QuickBuild's purchase price is $2999 (unlimited user)
- Cruise is a "software release management" product and includes many unrelated functionality besides the CI server functionality. For this reason, it fails from HLC 15.
- BuildForge standard edition has a price of $125,190.00 Therefore, it fails from HLC1
- Gump and CABIE do not have a security mechanism. Therefore they fail from HLC3.
- Cerberus is a continuous builder for Ruby not JAVA. Therefore it fails from HLC14.
- Control Tier is actually an application service management product. For this reason, it fails from HLC 15.
- CruiseControl.NET, CruiseControl.rb, Final Builder, Draco.NET and EasyCIS are not compatible with JAVA technology. Therefore they fail from HLC14.
- ElectricCommander is not compatible with the required browsers. Therefore, it fails from HLC12.
- JhBuild is developed for packaging modules; it is not a fully functional continuous integration server. Therefore, it fails from HLC15.
- OpenMake Meister is priced at $875 per named seat and $3,000 per concurrent seat.
- TeamCity has a price of $1999 (unlimited user)
- Tinderbox is more of a testing tool than a continuous integration server. Therefore, if fails from HLC15.

Results:

The tools that satisfy high level criteria: CruiseControl, Hudson, Continuum, LuntBuild, QuickBuild, AnthillPro, Automated Build Studio, Bamboo, Beebox, OpenMake Meister, OpenMake Mojo, ParaBuild, Pulse, TeamCity

Out of 30 tools, 14 passed, 16 failed.

2. Configuration Management Tool Area:

Table 16: Configuration management tool – criterion matching table (checklist)

	HLC ID																
Tool Name	**14**	**20**	**3**	**6**	**10**	**8**	**7**	**1**	**16**	**5**	**4**	**12**	**2**	**9**	**17**	**18**	**19**
AccuRev	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BitKeeper	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
ClearCase	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Synergy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Co-Op	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne
Perforce	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne
PureCM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Surround SCM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TFS	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Vault	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne
VSS	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
CVS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**X**	ne	ne
Aegis	✓	✓	✓	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Bazaar	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Darcs	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Mercurial	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Monotone	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
OpenCM	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Subversion	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Svk	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Vesta	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**X**	ne

✓: Criterion is successfully satisfied by the tool

X: Criterion is not satisfied by the tool

ne: Not Evaluated

Notes:

- Accurev license cost is $1,495 per user
- BitKeeper does not include a mechanism for remote access, the code must be held locally in order to work.
- ClearCase license cost is $4225 per floating license.
- Synergy license cost is $8,774 per floating license. Therefore, it fails from HLC1 when the number of developers in the department is considered.
- Perforce does not support directory versioning.
- Although users can version Java using TFS, it is not developed to work with Java.
- Although CVS can satisfy some HLCs by using open source plugins, it fails from HLC17 and HLC18.

Results:

The tools that satisfy high level criteria: Accurev, ClearCase, Synergy, PureCM, Surround SCM, Bazaar, Mercurial, Subversion, Svk

Out of 21 tools, 9 passed, 12 failed.

3. **Build Tool Area:**

Table 17: Build tool – criterion matching table (checklist)

Tool Name	21	14	13	6	10	8	1	12	2	4	5	9	7
Ant	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NAnt	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Maven	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Phing	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Rake	✓	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
XCode	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne	ne
Raven	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✓: Criterion is successfully satisfied by the tool

X: Criterion is not satisfied by the tool

ne: Not Evaluated

Notes:

- Nant is developed for .NET, C# building. Therefore, it fails from HLC 14.
- Phing is developed for PHP building. Therefore, it fails from HLC 14.
- XCode is an integrated development environment rather than a build tool and it also does not support Windows platforms. Therefore, it fails from HLC 22.

Results:

The tools that satisfy high level criteria: Ant, Maven, Raven

Out of 7 tools, 3 passed, 4 failed.

4.4 ACTIVITY 4: LOW LEVEL EVALUATION CRITERIA DEFINITION

In this activity, the evaluators formed the department's low level criteria for acquisition of the continuous integration tool(s). This type of criteria had to be more detailed than the previous class of high level criteria and required more extensive examination.

4.4.1 Step 1: Organizational Requirement Analysis

In the first step, internal requirements which resemble the needs stated by the deployer, developers and management were collected. They are given for each tool area below.

4.4.1.1 CI Server Tool Area:

- **Feedback options:** It is requested by the developers that the CI server should include some notification mechanism about the result of the builds it executes. There is a range of preferences about the type of the notification among developers so it is better to use a CI tool that offers various kinds of notifications. The popular types are email and RSS. The message is requested to include a link pointing to the build report or to the location where the error occurred in case of a build failure.

- **Extensibility:** Sometimes it becomes necessary to extend the tool to suit it to the organizational needs. For example, it may be necessary in the future to integrate the CI server with an in-house tool that the organization itself developed. For such cases, the CI server should have extension mechanisms.

- **Reliability:** It is requested by all stakeholders that the CI server be reliable. Reliability here is explained as the ability of the CI server to execute its processes correctly and repeatedly once they are constructed. Also, it is requested that in case of a failure the server should point the source of the error as intuitively as possible. If this is a build failure, the CI server should be able to show the version of the file or files that caused the build failure.

- **Longevity Prospects:** In cases the organization needs help for a problem with the tool, the vendor should be in place or the tool should still be in use by a community if it is open source software.

- **Application server requirement:** If the CI server application is distributed as a web application archive (war) file, it should be deployable to the Weblogic 9.2 server that is being used in the department. It is important to note that this requirement's subject is not the same as HLC13 which requires interoperability between the tools and the Weblogic server. This requirement is concerned about the installation support of the tool on Weblogic server.

- **Usability, ease of configuration:** The deployer in the department prefers a web interface for easy configuration. However he also requests to be able to modify the configuration via XML files since he is familiar with them owing to Ant. So, the evaluators concluded that a tool providing both of the interfaces is desirable.

- **Ease of installation**: It is required by the system administrators that the CI server should not have a complex installation procedure.

- **IDE integration:** The department is using Eclipse version 3.4.2 for JAVA development. Integration between the CI server and this version of Eclipse is requested both by the deployer and the developers.

4.4.1.2 Configuration Management Tool Area:

- **Remote access:** The developers use laptops for their work in the department. It is permitted to take the laptops home. However, some of the laptops had been stolen when they were outside the department in the past. The management now requests an SCM system where developers can access remotely without keeping any code on their laptops so no code can be stolen in case of a theft. Therefore, the SCM tool should provide a way for remote access to the repository while providing acceptable performance.

- **Folder level security:** It is required by the team leaders that the configuration management tool should support folder level security definitions. In the current system only project level security is supported.

- **Extensive documentation:** Previously, the department had difficulties in finding information for their tool. So, there is a requirement from developers that the configuration management system should provide adequate documentation that directs its users about its features. The documentation may be context-sensitive (such as the windows that come when the user presses F1 in tool window), web-based or in pdf format.

- **Complete Command Set and GUI:** To programmatically interact through shell, the tool should include all its functionality as available to be invoked from command line. However, it should also include graphical user interfaces that have the necessary functionality for the developers' daily SCM activities.

- **Robustness, Reliability:** It is requested by all stakeholders that the configuration management tool be reliable and robust. Reliability here is explained by the developers as: the tool should not fail in any case. Because a failure of the configuration management system generally means data loss and that data is usually source code.

- **Easy branching:** The configuration management tool should enable easy creation of branches. There are a lot of configuration management patterns that include branching mechanisms [15]. So opening a branch should be an easy task in the configuration management system. Also it should be easy to merge two separate branches.

- **IDE integration:** The department is using Eclipse version 3.4.2 for JAVA development. Integration between the configuration management tool and this version of Eclipse is requested by the developers.

4.4.1.3 Build tool area:

- **Java version:** The department is using JDK version 1.5.0_11. Applications being built are requested to run on the JVM included in this JDK. Therefore the build tool should support compiling and packaging applications using the interpreter that is included in this JDK.

- **IDE integration:** Integration between the build tool and Eclipse version 3.4.2 is requested by the developers.

- **Extensive Documentation:** Comprehensive documentation about the tool is requested by the developers in the department.

4.4.2 Step 2: External Analysis

In the external analysis step, the author performed a technology search to reveal additional requirements that are not explicitly stated by stakeholders but exist.

4.4.2.1 CI Server Tool Area:

- **Dashboard:** The CI server should be able to report its build results. This reporting may be from a web based dashboard that will show the users the state of recent and former builds. Color-coding and other visualization techniques may be employed in the dashboard for presentation purposes.

- **Labels:** It is requested that the CI server should give unique labels to its builds for identification purposes. Then the team can refer to a specific build by its label [86].

- **Project dependencies:** The projects in the environment may be interdependent to each other forcing a build order between them. So when the CI server starts the build of one of these projects, it should automatically trigger the build of the other project. [86]

- **Detailed bill-of-materials report:** The CI server should store comprehensive data on builds and related tasks to provide a detailed bill of materials that documents contents of each release for reproducibility and compliance management. Also, the CI server should keep track of which build produced which JAR and which build is using which version of a JAR. This is also called file fingerprinting.

- **Role-based user management:** The CI server should enable categorizing users in roles so that permissions may be granted rolewise.

- **Active Directory (LDAP) Authentication:** The department is using a Windows domain environment. The CI server's authentication mechanism should be able to retrieve username and password information from the LDAP server so that secondary efforts are not needed for user definitions and password changes are directly reflected to the system.

- **SCM Filtering:** Normally, the build should be triggered when a change in a source file occurs however it is necessary to exclude some type of files in this regard. For instance, it is not desired that the system starts a full build when a documentation file is changed in the SCM repository. Only code file changes should be taken into consideration.

- **Multiple SCM repository support:** The codebase that is being developed by the department may be dispersed into several SCM repositories. Therefore, the CI server should be capable of monitoring multiple repositories for change.

- **Historical graphs:** The CI server should be able to represent the build results belonging to a time frame in graphical format so that the trend of build results may be observed.

4.4.2.2 Configuration Management Tool Area:

- **Line-wise history tracking:** The configuration management system should be able to show which line is created or modified in which version and by whom in a specific text based versioned file. In other words, it should be able track the history of a file line by line.

- **Tracking uncommitted changes:** The configuration management system should have the ability to list the changes in the user's private workspace that he/she hasn't committed yet.

- **Per-file commit messages:** The configuration management system should be able to allow user to add a comment for each commit he/she makes.

- **Proper handling of binary files:** The configuration management tool should be able to handle binary files as well as text-based code files. Binary files should also be versioned if they are put under source control.

- **Data for managers:** The configuration management tool should provide statistical reports for the managers. For instance, a report can provide information to the manager about how many lines of code have been written during last month.

- **Optimistic locking:** Also called the copy-modify-merge model, this functionality enables developers to simultaneously work on the same resource which increases project pace. Currently, a strict locking model is being used which does not allow this. However, there is a possibly of conflict introduction into the file which both developers were working on. The configuration management tool should also be able to catch these kinds of conflicts and provide facilities for merging.

- **Branch and version labeling:** The configuration management tool should provide labeling features for versions and branches. This way, an important version or a set of versions can be identified.

No additional criteria were found from external sources concerning the build tool area.

4.4.3 Step 3: Criteria Formation for Each Tool Area

In this step, the author has consolidated all of the requirements found. Conflicts and overlappings between them were also resolved in this step. All of the criteria were documented in tabular format depicted by the proposed methodology. Priorities and types of the criteria were also determined and included in the tables. These are given for each tool area in Appendix A.

4.4.4 Step 4: Criteria Prioritization and Categorization

The criteria were prioritized as either normal or high in this step. Then, the normal priority criteria were categorized as functional, quality or supplier/community. The prioritization and categorization were also noted in the previous criteria tables. Tables 18-20 below show a summary of this information for each tool area.

Table 18: CI server tool criteria prioritization and categorization results

Criterion ID	Priority	Type
LLC1	High	NA
LLC2	Normal	Functional
LLC3	Normal	Quality
LLC4	High	NA
LLC5	High	NA
LLC6	Normal	Supplier/Community
LLC7	High	NA
LLC8	High	NA
LLC9	Normal	Quality
LLC10	Normal	Functional
LLC11	High	NA
LLC12	High	NA
LLC13	High	NA
LLC14	High	NA
LLC15	Normal	Functional
LLC16	Normal	Functional
LLC17	Normal	Functional
LLC18	High	NA
LLC19	Normal	Functional
LLC20	Normal	Functional

Table 19: Configuration management tool criteria prioritization and categorization results

Criterion ID	Priority	Type
LLC21	High	NA
LLC22	High	NA
LLC23	Normal	Quality
LLC24	Normal	Quality
LLC25	Normal	Quality
LLC26	Normal	Quality
LLC27	Normal	Functional
LLC28	High	NA
LLC29	Normal	Functional
LLC30	Normal	Functional
LLC31	Normal	Functional
LLC32	High	NA
LLC33	Normal	Functional
LLC34	High	NA
LLC35	High	NA
LLC36	High	NA
LLC37	Normal	Functional

Table 20: Build tool criteria prioritization and categorization results

Criterion ID	Priority	Type
LLC38	High	NA
LLC39	Normal	Quality
LLC40	Normal	Quality
LLC41	Normal	Quality
LLC42	Normal	Quality
LLC43	High	NA
LLC44	High	NA
LLC45	High	NA

Similar to high level criteria, these low level criteria and their priorities were validated by the assistant manager of the department as she was also a member of the evaluation team. As stated in the methodology, a group meeting with the stakeholders of the system might have also been arranged however since most of them were consulted during the formation of the criteria, we didn't consider this as necessary.

4.5 ACTIVITY 5: SCREENING

In the second screening, the author had to eliminate more tools which don't satisfy the criteria. According to the methodology, they had to evaluate the tools against the high priority low level criteria determined in the previous step. The two steps of this activity were executed in parallel ("Tool information gathering for each tool area" and "Candidate CASE tool selection for each tool area" steps). The screening results are demonstrated for each tool area in Tables 21-23.

4.5.1 CI Server Tool Area:

Table 21: CI server tool – criterion matching table 2 (checklist)

	LLC ID									
Tool Name	**1**	**4**	**5**	**7**	**8**	**11**	**12**	**13**	**14**	**18**
CruiseControl	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Hudson	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Continuum	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne
Luntbuild	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne
QuickBuild	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AnthillPro	✓	✓	✓	✓	**X**	ne	ne	ne	ne	ne
Automated Build Studio	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne
Bamboo	✓	✓	✓	✓	✓	✓	✓	✓	✓	**X**
Beebox	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne
OpenMake Meister	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne
OpenMake Mojo	**X**	ne	ne	ne	ne	ne	ne	ne	ne	ne
Parabuild	✓	✓	**X**	ne	ne	ne	ne	ne	ne	ne
Pulse	✓	✓	**X**	ne	ne	ne	ne	ne	ne	ne
TeamCity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✓: Criterion is successfully satisfied by the tool

X: Criterion is not satisfied by the tool

NE: Not Evaluated

Notes:

- Continuum, Luntbuild, Automated Build Studio, Beebox, OpenMake Meister and OpenMake Mojo do not have RSS support for notifications and no third plugin is found for this purpose. Therefore, they fail from LLC1.
- Anthill Pro is only configurable from Web; however LLC8 suggests that the tool should possess both the GUI and configuration file options. Therefore, it fails form LLC8.

- Parabuild is the only product of its producer: Viewtier systems. Although the tool offers good functionality, its market coverage is not big and this raises issues about the vendor stability criterion. This is also the case for Pulse which is the only product of Zutubi. Therefore, they fail from LLC5.
- Bamboo cannot differentiate SCM repository files when checking for changes. Therefore, it fails form LLC18.

Results:

Out of 14 tools, 4 of them have passed: CruiseControl, Hudson, QuickBuild, TeamCity

4.5.2 Configuration Management Tool Area:

Table 22: Configuration management tool – criterion matching table 2 (checklist)

	LLC ID						
Tool Name	**21**	**22**	**28**	**32**	**34**	**35**	**36**
AccuRev	**X**	ne	ne	ne	ne	ne	ne
ClearCase	✓	✓	✓	✓	✓	✓	✓
Synergy	**X**	ne	ne	ne	ne	ne	ne
PureCM	**X**	ne	ne	ne	ne	ne	ne
Surround SCM	**X**	ne	ne	ne	ne	ne	ne
Bazaar	**X**	ne	ne	ne	ne	ne	ne
Mercurial	**X**	ne	ne	ne	ne	ne	ne
Subversion	**X**	ne	ne	ne	ne	ne	ne
Svk	**X**	ne	ne	ne	ne	ne	ne

✓: Criterion is successfully satisfied by the tool

X: Criterion is not satisfied by the tool

NE: Not Evaluated

Notes:

- Accurev, Synergy, PureCM, Surround SCM, Bazaar, Mercurial, Subversion and Svk do not have remote access mechanisms other than their web clients. LLC21 states that a mechanism is requested to enable working in a fat client that does not hold code locally. A remote sandbox model is possible with some of these tools however this is not suitable for everyday use.

Results:

Out of 9 tools, 1 of them has passed: ClearCase.

4.5.3 Build Tool Area:

Table 23: Build tool – criterion matching table 2 (checklist)

	LLC ID			
Tool Name	**38**	**43**	**44**	**45**
Ant	✓	✓	✓	✓
Maven	✓	✓	✓	✓
Raven	**X**	ne	ne	ne

✓: Criterion is successfully satisfied by the tool

X: Criterion is not satisfied by the tool

NE: Not Evaluated

Notes:

- Raven build scripts are based on Ruby not XML. Developers in the department are not familiar with Ruby. Therefore Raven fails from criterion LLC37.

Results:

Out of 3 tools, 2 of them passed: Ant and Maven.

4.6 ACTIVITY 6: EVALUATION AND COMPARISON

In this activity, the tools that pass the screening activity were evaluated according to their level of support for the normal priority low level criteria. The evaluation was performed for each tool area in two steps.

4.6.1 Step1: Assessment method determination for each criterion

The assessment method of the low level criteria was determined in this step. Possible assessment methods given by the methodology were: official tool documentation lookup, case study on tool evaluation copy, visiting an existing user group, demonstration or information request from the supplier, online tool newsgroup or forum search, tool reference search.

More than one method was used for evaluation of some criteria as this was suggested by the methodology. Assessment method determination results for each tool area are given in Tables 24-26.

Table 24: Criterion – Assessment method table for CI server tool area

Criterion ID	Criterion Name	Assessment Method
LLC2	Extension mechanisms	Official tool documentation lookup
LLC3	Robust working	Visiting an existing user group Official tool documentation lookup Online tool newsgroup or forum search
LLC6	Vendor and tool reputation	Tool reference search
LLC9	Ease of installation	Case study on tool evaluation copy
LLC10	Eclipse integration	Official tool documentation lookup
LLC15	File fingerprinting support	Official tool documentation lookup Case study on tool evaluation copy
LLC16	Role-based user management	Official tool documentation lookup
LLC17	LDAP authentication	Official tool documentation lookup
LLC19	Multiple SCM repository support	Official tool documentation lookup Case study on tool evaluation copy
LLC20	Graphical build trends	Official tool documentation lookup Case study on tool evaluation copy

Table 25: Criterion – Assessment method table for configuration management tool area

Criterion ID	Criterion Name	Assessment Method
LLC23	Documentation	Official tool documentation lookup
LLC24	Extensive command set	Official tool documentation lookup Case study on tool evaluation copy Demonstration or information request from the supplier Online tool newsgroup or forum search
LLC25	Extensive graphical interfaces	Official tool documentation lookup Case study on tool evaluation copy Demonstration or information request from the supplier
LLC26	Robustness	Visiting an existing user group Case study on tool evaluation copy Online tool newsgroup or forum search Demonstration or information request from the supplier
LLC27	Branching abilities	Official tool documentation lookup Case study on tool evaluation copy Demonstration or information request from the supplier
LLC29	Line wise history tracking	Official tool documentation lookup Case study on tool evaluation copy
LLC30	Uncommitted data indication	Official tool documentation lookup Case study on tool evaluation copy
LLC31	Per-file commit messages	Official tool documentation lookup
LLC33	Reporting options	Official tool documentation lookup Demonstration or information request from the supplier
LLC37	Embedded database	Official tool documentation lookup

Table 26: Criterion – Assessment method table for build tool area

Criterion ID	Criterion Name	Assessment Method
LLC39	Easy installation	Case study on tool evaluation copy
LLC40	Easy project configuration	Official tool documentation lookup Case study on tool evaluation copy
LLC41	Shallow learning curve	Official tool documentation lookup Demonstration or information request from the supplier
LLC42	Complete documentation	Official tool documentation lookup

4.6.2 Step2: Assessment and Comparison for Each Criterion

Actual assessments were performed in this step using the methods previously stated. The assessment details were documented in tables. Also, the comparisons were made according the Saaty's verbal scale [10] as depicted in the methodology. The tables showing the assessment results for each tool area are given in Appendix B.

4.7 ACTIVITY 7: RANKING AND SELECTION

In this activity, the intent is to conclude the selection process with the necessary computations described by the methodology. These computations were performed in the first step of the activity which includes the application of the AHP methodology. In the second step, the evaluators announced the final decisions after reconsidering some issues detailed by the methodology.

4.7.1 Step 1: AHP Application

Six steps which form this activity were applied for the two tool areas sequentially. These tool areas were the CI Server tool area and the build tool area. Since configuration management tool area only has one tool, this area was not involved in the computations.

Moreover, the computations in this step were performed according to the manual approximation technique detailed by the methodology. Although the methodology states the possibility of utilizing the “Expert Choice” tool, this investment was not made.

4.7.1.1 FOR CI SERVER TOOL AREA:

i. Hierarchical problem model construction

The four level hierarchy of the problem was defined in Figure 18.

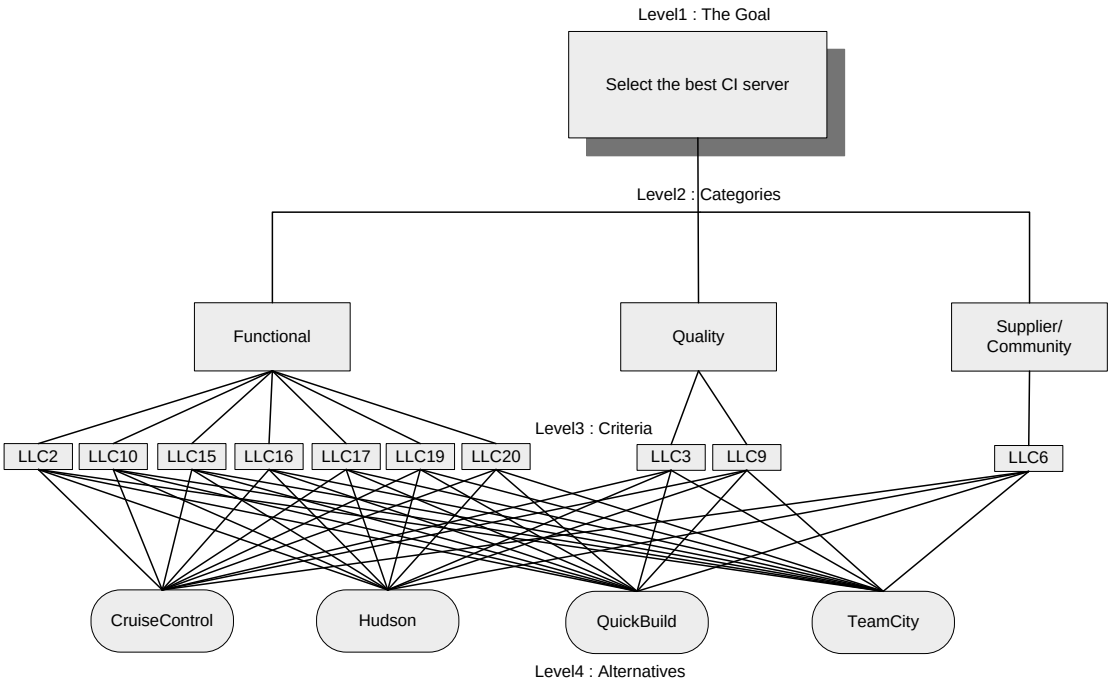

Figure 18: AHP problem hierarchy

ii. Derivation of ratio scale priorities for the categories

The relative priorities of the categories were computed according to the methodology. Results are given in Table 27.

Table 27: Category comparison table

	Functional	Quality	Supplier/Community	Approximate Weight
Functional	1	3	5	9.000 (.570)
Quality	1/3	1	4	5.333 (.338)
Supplier/Community	1/5	1/4	1	1.450 (.092)
Total				15.783

iii. Derivation of ratio scale priorities for the criteria

The computed relative priorities of the criteria for each category are given in Tables 28-33. Functional criteria table and Saaty's verbal scale table is included for quick reference.

- **For the functional category:**

Criteria List:

Table 28: Functional criteria table

LLC2	Extension mechanisms
LLC10	Eclipse integration
LLC15	File fingerprinting support
LLC16	Role based user management
LLC17	LDAP authentication
LLC19	Multiple SCM repository support
LLC20	Graphical build trends

Table 29: Saaty's verbal scale table [10]

Expressed Judgment of Preference	Numerical Value
Extremely preferred	9
Between very strongly and extremely	8
Very strongly preferred	7
Between strongly and very strongly	6
Strongly preferred	5
Between moderately and strongly	4
Moderately preferred	3
Between equally and moderately	2
Equally preferred	1

Table 30: Functional criteria comparison table

	LLC2	LLC10	LLC15	LLC16	LLC17	LLC19	LLC20	Approximate Weight
LLC2	1	1/7	1/3	1/7	1/2	1/4	1/3	2.702 (.031)
LLC10	7	1	5	2	5	3	5	28.000 (.322)
LLC15	3	1/5	1	3	2	1/3	1	10.533 (.121)
LLC16	7	1/2	1/3	1	5	2	2	17.833 (.205)
LLC17	2	1/5	1/2	1/5	1	1/4	1/3	4.483 (.051)
LLC19	4	1/3	3	1/2	4	1	1	13.833 (.159)
LLC20	3	1/5	1	1/2	3	1	1	9.700 (.111)
Total								87.084

- **For the quality category:**

Criteria List:

Table 31: Quality criteria table

LLC3	Robust working
LLC9	Ease of installation

Table 32: Quality criteria comparison table

	LLC3	**LLC9**	**Approximate Weight**
LLC3	1	7	8.000 (.875)
LLC9	1/7	1	1.143 (.125)
Total			9.143

- **For the supplier/community category:**

Criteria List:

Table 33: Supplier/community criterion

LLC6	Vendor and tool reputation

Since there is only 1 criterion in this category, its weight is directly 1.

iv. Derivation of ratio scale priorities for the alternatives

The comparisons previously defined in the evaluation activity were utilized in this step to calculate the relative priorities of the alternatives according to the scheme given by the methodology. These calculations are organized in synthesis tables which are given in Appendix C.

v. Consistency ratio estimation

In this step, the results of the previous comparisons were evaluated to determine their consistency. Consistency ratio indexes were computed according to AHP as described in the methodology. All the comparisons were found to be consistent since their index values were lower than 0.1. The details of the computations are given in the tables which were constructed according to the template given in the methodology. These tables can be seen in Appendix D.

vi. Overall priority ranking

In this step, all the relative priorities found for the CI server tool area were combined to obtain the final ranking of the tools. The ranking calculations performed are given in the Tables 34-37.

- **Functional Category:**

Table 34: Ranking table for functional category

	Functional							
Ranking	**LLC2**	**LLC10**	**LLC15**	**LLC16**	**LLC17**	**LLC19**	**LLC20**	**TOTALS**
CruiseControl	0.031	0.322	0.121	0.205	0.051	0.159	0.111	
	X	X	X	X	X	X	X	
	0.501	0.308	0.042	0.042	0.036	0.308	0.054	0.185
Hudson	0.031	0.322	0.121	0.205	0.051	0.159	0.111	
	X	X	X	X	X	X	X	
	0.219	0.308	0.592	0.592	0.321	0.077	0.107	0.339
QuickBuild	0.031	0.322	0.121	0.205	0.051	0.159	0.111	
	X	X	X	X	X	X	X	
	0.062	0.077	0.214	0.214	0.321	0.308	0.520	0.220
TeamCity	0.031	0.322	0.121	0.205	0.051	0.159	0.111	
	X	X	X	X	X	X	X	
	0.219	0.308	0.153	0.153	0.321	0.308	0.319	0.257

CI Server Tool Ranking For the Functional Criteria:

1. Hudson
2. TeamCity
3. QuickBuild
4. CruiseControl

- **Quality Category:**

Table 35: Ranking table for quality category

	Quality		
Ranking	**LLC3**	**LLC9**	**TOTALS**
CruiseControl	0.875x0.449	0.125x0.333	0.435
Hudson	0.875x0.235	0.125x0.333	0.248
QuickBuild	0.875x0.235	0.125x0.167	0.227
TeamCity	0.875x0.082	0.125x0.167	0.093

CI Server Tool Ranking For the Quality Criteria:

1. CruiseControl
2. Hudson
3. QuickBuild
4. TeamCity

- **Supplier/Community Category**

Table 36: Ranking table for supplier/community category

	Supplier/Community	
Ranking	**LLC6**	**TOTALS**
CruiseControl	1x0.483	0.483
Hudson	1x0.261	0.261
QuickBuild	1x0.070	0.070
TeamCity	1x0.186	0.186

CI Server Tool Ranking For the Supplier/Community Criteria:

1. CruiseControl
2. Hudson
3. TeamCity
4. QuickBuild

- **Overall Ranking**

Table 37: Overall ranking table for CI server tool area

	Overall			
Ranking	**Functional**	**Quality**	**Supplier/Community**	**TOTALS**
CruiseControl	0.185x0.570	0.435x0.338	0.483x0.092	0.297
Hudson	0.339 x0.570	0.248 x0.338	0.261 x0.092	0.301
QuickBuild	0.220 x0.570	0.227 x0.338	0.070 x0.092	0.209
TeamCity	0.257 x0.570	0.093 x0.338	0.186 x0.092	0.195

Overall CI Server Tool Ranking:

1. Hudson
2. CruiseControl
3. QuickBuild
4. TeamCity

4.7.1.2 FOR BUILD TOOL AREA:

Same steps performed for the CI server tool area were repeated for the build tool area in this section.

i. Hierarchical problem model construction

The four level hierarchy of the problem was defined in Figure 19 below.

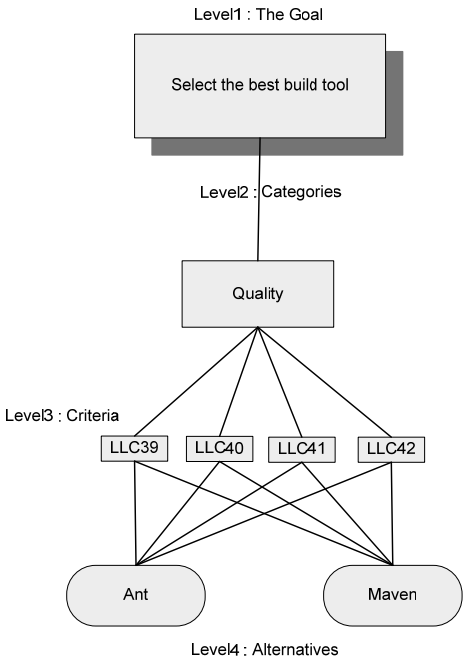

Figure 19: AHP problem hierarchy

ii. Derivation of ratio scale priorities for the categories

Since, there is only Quality category; its relative priority is directly one.

iii. Derivation of ratio scale priorities for the criteria

- **For the quality category:**

Criteria List:

Table 38: Quality criteria table

LLC39	Easy installation
LLC40	Easy project configuration
LLC41	Shallow learning curve
LLC42	Complete documentation

Table 39: Quality criteria comparison table

	LLC39	**LLC40**	**LLC41**	**LLC42**	**Approximate Weight**
LLC39	1	1/5	1/5	1/5	1.600 (.057)
LLC40	5	1	3	3	12.000 (.425)
LLC41	5	1/3	1	1	7.333 (.259)
LLC42	5	1/3	1	1	7.333 (.259)
Total					28.266

iv. Derivation of ratio scale priorities for the alternatives

The synthesis tables constructed for this section are given in Appendix C.

v. Consistency ratio estimation

Consistency of the comparisons was very obvious so we didn't need to check for consistency.

vi. Overall priority ranking

Since there was only one category to evaluate, the category based ranking was equal to the overall ranking.

Table 40: Overall ranking table for build tool area

	Overall Ranking (equals Quality Ranking)				
Ranking	**LLC39**	**LLC40**	**LLC41**	**LLC42**	**TOTALS**
Ant	0.057x0.500	0.425x0.200	0.259x0.800	0.259x0.500	0.450
Maven	0.057x0.500	0.425x0.800	0.259x0.200	0.259x0.500	0.550

Overall Build Tool Ranking:

1. Maven
2. Ant

4.7.2 Step2: Final Selection

In this step, the evaluators made their final decisions about the CASE tool acquisition considering the computed rankings. The conclusions arrived by the evaluators are given for each tool area below.

- *For the CI server tool area*

It can be seen from the ranking that Hudson and CruiseControl have taken the lead against our commercial candidates, QuickBuild and TeamCity. However, the difference between Hudson and CruiseControl is not very much which means that they both satisfy the organizational requirements at a nearly equal degree. Hudson has relatively better functionality but newer in market compared to CruiseControl which is more stable and has a large user community.

In accordance with the results, the evaluators choice was using Hudson as the CI server tool and they also stated that if Hudson had been a little behind CruiseControl they would have again selected it since they are enthusiastic about trying the newer tools.

- *For the configuration management tool area*

Since ClearCase was the only tool that satisfied the screening criteria, the obvious result here is to use ClearCase instead of VSS that is currently being used. This is mostly because of the management's desire about not keeping the code in the developers' computers since this was the criterion that ClearCase was natively providing and the other tools were not. The evaluators are also glad about the decision because ClearCase also proved well in the mentioned normal priority low level criteria and the tool will be supported by IBM.

- *For the build tool area*

In this area, Maven has taken the lead over Ant by taking ten percent higher weight. If we look at the ranking table, we can see that this difference is mainly due to the easy project configuration ability of Maven. However, the evaluators think that this difference is not enough to cover the replacement costs of Ant. The department's familiarity with Ant and the tool's good integration with Hudson and ClearCase further supplement the decision of the evaluators that is keeping Ant.

Moreover, the cost of the solution is further evaluated and decided to be in the boundaries defined by the management and tools' integrations between each other was decided to be sufficiently good.

So, the final set of tools that will be used in the department's continuous integration process was decided to be Hudson, ClearCase and Ant.

4.7.3 Discussion about the case study:

After finalizing the application of the proposed methodology, the effectiveness of the case study in regard to the research questions posed at the beginning of Chapter 4 can be assessed.

First of all, the methodology is found feasible to apply in the subject department of the institution since all of the activities could be performed as defined in the methodology which in turn resulted in a meaningful result. Moreover, all of the activities of the methodology have been finished in the timeframe that was dedicated for the process that is one and a half month even though the evaluators were also performing their usual duties. Since the number of the alternatives were high for each tool area, we believe that the screening activities have considerably decreased the time and effort required for comparisons.

Because of the continuous literature search that has been made during the course of the methodology, the knowledge of the evaluators about the continuous integration process and the tools developed for it have been substantially enhanced. Also, some problems in the department's development procedures were revealed during the elicitation of internal requirements.

After finishing the application, the senior developer who was a member of the evaluation team commented about the process as being comprehensive enough to provide a solid decision among the CASE tool solutions. After making discussions with some of the stakeholders, it has been found that another criterion that necessitates an integration with a specific workflow system being used in the department would have also been added. On the other hand, it is observed that the tools in the final selection already provide this integration. Also, two low level criteria that we determined as normal priority would have been selected as high priority regarding the feedback given by some developers. We have also concluded that such a change would not the affect the final ranking.

Moreover, the AHP technique was found very valuable by the involved employees and evaluators mostly due to the pairwise comparison logic and consistency checking facility provided by the method. They told that the technique ensured examination of every detail in a consistent manner and the usage of a verbal scale instead of exact numerical scorings made the comparisons more accurate. The managers of the department also acknowledged the organizational knowledge enhancement resulted from the methodology and approved the proposed CASE tool combination.

It is concluded that if the CASE tool acquisition efforts were conducted without using the methodology, the same tool combination would not have been selected. For example, the authorized people in the department were unaware of the existence of CI server tools like Hudson and they were considering to delegate this job to a worker to be performed manually. For the configuration management area, they were considering to replace VSS with ClearCase as an option however they were unsure whether better possibilities exist. Lastly, for the build area, they didn't know the improvements in the technology and emergence of tools like Maven.

So, we can deduce the result that the institution has gained the expected benefits from applying the proposed methodology because of the reasons stated. The organization would have also employed the process proposed by ISO [5] but the success is questionable because the ISO 14102 standard does not include a criteria set for the continuous integration process. Likewise, other studies on this field which are limited in number do not seem to offer the detailed systematic approach proposed by this study to the best of our knowledge.

Finally, it is observed that the evaluators and the management have a high level of confidence for the tool combination resulting from application of the methodology. This is because all the decisions that resulted in forming the final tool combination were made by internal evaluators and based on either organizational requirements or technological observations. The tool combination proposed by the methodology will actually be acquired and put into use before the second quarter of 2010.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Utilization of CASE tools in software development projects is continuously increasing. This also contributes to an increase in the number of producer firms and directs them to offer more tools with more capabilities. So, the CASE tools in the market are increasing in terms of number and in terms of the functionalities they offer. Organizations on the other hand are looking for solutions that will increase product quality and decrease development costs. However, these benefits may not be realized because of a wrong tool selection; a tool that does not fit into the organizational context and does not provide the requirements of the organization. This may be due to a nonprocedural (ad hoc) decision about the tool to be acquired. However, for such an acquisition that can affect overall development progress, a more structured decision model should be used.

In this thesis, a systematic methodology for CASE tool evaluation and selection is presented to address this need. This methodology is designed to be used in situations where there are many tools to assess in a short timescale. Moreover, the organisation, internal consistency and applicability of the methodology is exercised on a case study.

The proposed methodology can be distinguished from other work in the field because it encompasses the combination of the aspects given below:

- End to end process definition: The methodology starts from identifying the problem of the organization that led to the CASE tool acquisition attempt and continues until a tool or a combination of tools is selected which solve this problem.

- Criteria formation from internal and external requirements: The methodology does not propose a standard criteria set for CASE tool selection. Instead, criteria elicitation from the stakeholders is defended since this kind of effort will result in criteria that represent organizational needs. Also, analysis of external resources is performed to catch the overlooked requirements and features that will be needed in the long run.

- Question sets for easing criteria elicitation: To help criteria elicitation, the proposed methodology includes question sets. Answering to these questions will aid evalutors to discover some criteria.

- Progressive screening: When the high number of CASE tools and their functionalities are considered, it can easily be deduced that a vast amount effort would be needed if we directly apply the comparison and ranking over all the candidates. Therefore, the proposed methodology presents a two stage screening operation to reduce the number of candidates that will be evaluated deeply.

- Tool area and tool combination concepts: The proposed methodology acknowledges the fact that a CASE tool combination may be the best solution for the organization instead of a single CASE tool. To address this concern, the concept of tool areas is proposed within the methodology.

- Self-learning aspects: The activities in the methodology not only results in a tool selection but also enhances the team's knowledge of the organizational requirements and CASE technology.

However, the proposed methodology also have some limitations which are given below.

Limitations

- The methodology aids in the selection of the most suitable tool or tools for the intended software development practice however the CASE technology itself cannot cover all the aspects of a practice. Since software development is a social effort, it is the users' responsibility to utilize the tools properly as required by the practice they are performing.

- A wrong CASE tool selection may cause failure of an entire business, therefore a more formal methodology might be needed. However, this is a rare situation considering the usage scope of CASE tools. So, such an endeavor which requires more effort to apply is not required in practical cases.

- The proposed methodology is only exercised in a single case study that includes "continuous integration" practice as the subject. Application of the methodology for different practices may reveal some shortcomings not anticipated during the design.

Case Study Analysis

The case study demonstrated that the methodology is appropriate for selecting a tool to be used in the continuous integration process of the subject department. During the application of the methodology, the evaluators gained knowledge about the organizational needs and trends in today's CASE technology. The screening phases of the methodology have eliminated many candidates and decreased the effort that is needed for evaluation while enabling the work to be completed in the determined timescale (one and a half month). After the evaluations and comparisons, a tool combination is selected which gained management approval due to realistic reasonings provided by the methodology. Selection of the same combination without utilizing the methodology is found unlikely by the evaluators since some of the tools and technologies were not known before.

So, we can conclude that the application of the methodology was successful and provided the expected benefits. However, we understood from the feedbacks that we would get better coverage if the criteria determination phases of the methodology would have been conducted in a group setting involving all the stakeholders instead of just the evaluators. Also, according to the feedback we received, the integration related requirements would have been considered more deeply.

Future Work

Following topics may be covered as future work:

- A tool may be developed to handle and direct all the activities involved in the methodology.

- Implications of group decision making which is required at some points in the methodology may be investigated more throughly.

- Analytical Hierarchy Process (AHP) is chosen for its good coverage of qualitative decisions and group settings. However, other ranking techniques of Multi Criteria Decision Making (MCDM) may also be evaluated for suitability.

REFERENCES

1. Kontio, J. 1996. A Case Study in Applying a Systematic Method for COTS Selection. Proceedings of the 18th international conference on Software engineering, 201 – 209.

2. Fuggetta, A. 1993. A Classification of CASE Technology. Computer, 26(12): 25 – 38.

3. Firth, R., Mosley, V., Pethia, R., Roberts, L., Wood, W. 1987. A Guide to the Classification and Assessment of Software Engineering Tools (CMU/SEI-87-TR-10).

4. Forman, E. H., Saul, I. G. 2001. The analytical hierarchy process—an exposition Operations Research 49(4): 469–487.

5. ISO 14102:2008. Information technology -- Guideline For The Evaluation and Selection of CASE Tools.

6. Lundell, B., Lings, B. 2002. Comments on ISO 14102: the Standard for CASE-tool Evaluation. Computer Standards & Interfaces, 24(5), 381 – 388.

7. Blanc, A. L., Korn W. M. 1992. A Structured Approach to The Evaluation and Selection of CASE Tools. Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's, 1064 - 1069.

8. Jadhav, A. S., Sonar, R. M. 2009. Evaluating and Selecting Software Packages: A Review. Information and Software Technology 51 555–563.

9. Comella-Dorda S., Dean, J., Lewis, G., Morris, E., Oberndorf, P., Harper, E. A. 2004. Process for COTS Software Product Evaluation. Technical Report CMU/SEI-2003-TR-017.

10. Saaty, T.L., 1980. The Analytic Hierarchy Process. McGraw-Hill, New York.

11. Jayaswal, B. K., Patton, P. C. with Forman, E. H. 2007. The Analytic Hierarchy Process (AHP) in Software Development. Prentice Hall. Retrieved November 29, 2009 from METU library: http://my.safaribooksonline.com/9780132351355

12. Expert Choice, http://www.expertchoice.com/ Last accessed: November 28, 2009

13. Maccari, A., Riva, C. 2000. Empirical Evaluation of CASE Tools Usage at Nokia. Empirical Software Engineering, 5(3), 287 – 299.

14. CruiseControl, http://cruisecontrol.sourceforge.net/main/install.html Last accessed: November 28, 2009

15. Berczuk, S., Appleton, B. 2003. Software Configuration Management Patterns. ISBN: 0201741172 2003 Addison-Wesley

16. CruiseControl, http://cruisecontrol.sourceforge.net/main/plugins.html Last accessed: November 28, 2009

17. Hudson, http://wiki.hudson-ci.org/display/HUDSON/Extend+Hudson Last accessed: November 28, 2009

18. QuickBuild, http://wiki.pmease.com/display/qb2/Plugin+Management Last accessed: November 28, 2009

19. TeamCity, http://www.jetbrains.net/confluence/display/TCD4/Developing+TeamCity+Plugins Last accessed: November 28, 2009

20. QuickBuild,
http://track.pmease.com/browse/QB?report=com.atlassian.jira.plugin.system.project:openissues-panel Last accessed: November 28, 2009

21. JavaOne Conference 2008,
http://java.sun.com/javaone/sf/2008/articles/2008dukeschoiceawards.jsp
Last accessed: November 28, 2009

22. CruiseControl, http://code.google.com/p/cruisecontrol-eclipse-plugin/
Last accessed: November 28, 2009

23. Hudson, http://code.google.com/p/hudson-eclipse/
Last accessed: November 28, 2009

24. ISO / IEC 12207:2008. Systems and software engineering – Software life cycle processes.

Continuous Integration Server Tools:

25. CruiseControl, http://cruisecontrol.sourceforge.net
Last accessed: November 28, 2009

26. Hudson, http://hudson-ci.org/
Last accessed: November 28, 2009

27. Continuum, http://continuum.apache.org/
Last accessed: November 28, 2009

28. Luntbuild, http://luntbuild.javaforge.com/
Last accessed: November 28, 2009

29. QuickBuild, http://www.pmease.com/
Last accessed: November 28, 2009

30. Cruise, http://www.thoughtworks-studios.com/cruise-release-management
Last accessed: November 28, 2009

31. BuildForge, http://www.ibm.com/software/awdtools/buildforge
Last accessed: November 28, 2009

32. Anthill Pro, www.anthillpro.com
Last accessed: November 28, 2009

33. Gump, http://gump.apache.org
Last accessed: November 28, 2009

34. Automated Build Studio, http://www.automatedqa.com/products/abs
Last accessed: November 28, 2009

35. Bamboo, http://www.atlassian.com/software/bamboo/
Last accessed: November 28, 2009

36. Beebox, http://www.beebox.ca/en/home.html
Last accessed: November 28, 2009

37. Cabie, http://cabie.tigris.org/
Last accessed: November 28, 2009

38. Cerberus, http://cerberus.rubyforge.org/
Last accessed: November 28, 2009

39. CruiseControl.NET,
http://confluence.public.thoughtworks.org/display/CCNET/Welcome+to+Cruise Control.NET
Last accessed: November 28, 2009

40. CruiseControl.rb, http://cruisecontrolrb.thoughtworks.com/
Last accessed: November 28, 2009

41. Control Tier, http://open.controltier.org/wiki/Main_Page
Last accessed: November 28, 2009

42. Draco.NET, http://draconet.sourceforge.net/
Last accessed: November 28, 2009

43. EasyCIS, http://www.easycis.eu/web/features.aspx
Last accessed: November 28, 2009

44. Electric Commander, http://www.electric-cloud.com/products/electriccommander.php
Last accessed: November 28, 2009

45. Final Builder, www.FinalBuilder.com
Last accessed: November 28, 2009

46. InstallAce, http://www.installace.com/Default.aspx
Last accessed: November 28, 2009

47. OpenMake Meister, http://www.openmakesoftware.com/meister-7.0/
Last accessed: November 28, 2009

48. OpenMake Mojo, http://www.openmakesoftware.com/mojo-max-info/
Last accessed: November 28, 2009

49. Parabuild, http://www.viewtier.com/index.htm
Last accessed: November 28, 2009

50. Pulse, http://zutubi.com/
Last accessed: November 28, 2009

51. TeamCity, http://www.jetbrains.com/teamcity/
Last accessed: November 28, 2009

52. Team Foundation Server, http://msdn.microsoft.com/en-us/teamsystem/default.aspx
Last accessed: November 28, 2009

53. Tinderbox, http://www.mozilla.org/tinderbox.html
Last accessed: November 28, 2009

End of continuous integration server tools

Configuration Management Tools:

54. Accurev, http://www.accurev.com/accurev.html
Last accessed: November 29, 2009

55. Bitkeeper, http://www.bitkeeper.com/Products.html
Last accessed: November 29, 2009

56. IBM Rational Clearcase, http://www-01.ibm.com/software/awdtools/clearcase/
Last accessed: November 29, 2009

57. IBM Rational Synergy, http://www-01.ibm.com/software/awdtools/synergy/
Last accessed: November 29, 2009

58. Co Op, http://www.relisoft.com/co_op/index.htm
Last accessed: November 29, 2009

59. Perforce, http://www.perforce.com/
Last accessed: November 29, 2009

60. PureCM, http://www.purecm.com/
Last accessed: November 29, 2009

61. SourceAnywhere, http://www.componentsource.com/products/sourceanywhere-standalone/index.html
Last accessed: November 29, 2009

62. Surround SCM, http://www.seapine.com/surroundscm.html
Last accessed: November 29, 2009

63. Team Foundation Server, http://msdn.microsoft.com/en-us/teamsystem/default.aspx
Last accessed: November 29, 2009

64. Vault, http://www.sourcegear.com/vault/
Last accessed: November 29, 2009

65. VSS, http://msdn.microsoft.com/en-us/library/3h0544kx%28VS.80%29.aspx
Last accessed: November 29, 2009

66. CVS, http://www.nongnu.org/cvs/
Last accessed: November 29, 2009

67. Aegis, http://aegis.sourceforge.net/
Last accessed: November 29, 2009

68. Bazaar, http://bazaar.canonical.com/en/
Last accessed: November 29, 2009

69. Darcs, http://ostatic.com/darcs
Last accessed: November 29, 2009

70. Mercurial, http://mercurial.selenic.com/
Last accessed: November 29, 2009

71. Monotone, http://www.monotone.ca/
Last accessed: November 29, 2009

72. Open CM, http://www.opencm.org/
Last accessed: November 29, 2009

73. Subversion, http://subversion.tigris.org/
Last accessed: November 29, 2009

74. Svk, http://bestpractical.com/svk/
Last accessed: November 29, 2009

75. Vesta, http://www.vestasys.org/
Last accessed: November 29, 2009

End of configuration management tools

Build Tools:

76. Ant, http://ant.apache.org
Last accessed: November 29, 2009

77. NAnt, http://nant.sourceforge.net/
Last accessed: November 29, 2009

78. Maven, http://maven.apache.org
Last accessed: November 29, 2009

79. Phing, http://phing.info/trac/
Last accessed: November 29, 2009

80. Rake, http://rubyforge.org/projects/rake/
Last accessed: November 29, 2009

81. XCode, http://developer.apple.com/tools/xcode/
Last accessed: November 29, 2009

82. Raven, http://rubyforge.org/projects/raven/
Last accessed: November 29, 2009

End of build tools

83. Post, G., Kagan, A., Keim, R. T. 1998. A Comparative Evaluation of CASE Tools. Journal of Systems and Software, 44(2), 87 – 96.

84. Lai, V. S., Trueblood, R. P., Wong, B. K. 1997. Software Selection: A Case Study of The Application of the Analytical Hierarchical Process to The Selection of Multimedia Authoring System Information & Management 36 (1999) 221±232.

85. Dyer, R. F., Forman, E. H., 1992. Group Decision Support With the Analytic Hierarchy Process. Decision Support Systems, 8(2), 99 – 124.

86. Duvall, P. 2006. Automation For The People: Choosing A Continuous Integration Server. http://www.ibm.com/developerworks/java/library/j-ap09056/index.html
Last accessed: November 15, 2009

87. Gartner Group, New Bern, NC. http://www.thegartnergroup.com/
Last accessed: November 20, 2009

88. Ovum, Boston, MA. http://www.ovum.com/
Last accessed: November 20, 2009

89. Kitchenham, B. 1996. DESMET: A Methodology For Evaluating Software Engineering Methods and Tools. Computing & Control Engineering Journal, 8(3), 120 – 126.

90. Sadler, C., Kitchenham, B. A. 1996. ACM Sigsoft Evaluating Software Engineering Methods and Tool Part4: The Influence of Human Factors. ACM SIGSOFT Software Engineering Notes, 21(5), 11 – 13.

91. Ballestero, E. 1998. Multiple Criteria Decision Making and its Applications to Economic Problems Springer ISBN: 978-0792382386.

92. Figueira, J. 2005. Multiple Criteria Decision Analysis: State of the Art Surveys (International Series in Operations Research & Management Science). Springer ISBN: 978-0-387-23067-2.

93. CASE History, http://it.toolbox.com/wiki/index.php/History_of_CASE Last accessed: January 22, 2010

94. Candrlic, S., Pavlic, M., Poscic, P. 2007. A comparison and The Desirable Features of Version Control Tools Information Technology Interfaces. Issue 25-28 June 2007 Page(s):121 – 126.

95. Kunda D. 2003. STACE: Social Technical Approach to COTS Software Evaluation. Component-Based Software Quality, LNCS 2693, 64 – 84.

96. Vessey, I., Sravanapudi, A. 1992. Evaluation of Vendor Products: CASE Tools in Support of Work Groups. System Sciences. Proceedings of the Twenty-Fifth Hawaii International Conference, 4, 420 – 431.

97. Bruckhaus, T., Madhavji, N. H., Janssen I., Henshaw J. 1996. The Impact of Tools on Software Productivity. IEEE Software, 13(5), 29 – 38.

98. Lundell, B., Lings, B. 2004. On Understanding Evaluation of Tool Support for IS Development. Australasian Journal of Information Systems, 12(1), 39 – 53.

99. 1993 IEEE Std 1209-1992. Recommended Practice for the Evaluation and Selection of CASE Tools.

100. Sodhi, J. 1991. Software Engineering: Methods, Management, and CASE Tools, McGraw-Hill, Blue Ridge Summit, Pa.

101. Pressman, R.S. 1992. Software Engineering - A Practitioner's Approach, McGraw-Hill, New York.

102. Forte, G., McCulley, K. 1991. CASE Outlook: Guide to Products and Services, CASE Consulting Group, Lake Oswego, Ore.

103. Sommerville, I. 1992. Software Engineering, Addison-Wesley, Reading, Mass.

104. Thomas, I., Nejmeh, B.A. 1992. Definition of Tool Integration for Environments, IEEE Software, Vol. 9, No. 2, Mar., pp. 29-35.

105. Brown, A.W., Wallnau, K.C. 1996. A Framework for Evaluating Software Technology, IEEE Software, Vol. 13 (5), 39-49

106. ISO 1998 Information Technology - Software Product Evaluation - Part 5: Process for evaluators, ISO/IEC 14598-1:1998(E), 1998-07-01

107. Van Reeken, A.J., Trienekens, J.J.M. 1992. The Practical Importance of Methods and Case Tools: Results of Empirical Research, MERIT 92-015, Maastricht Economic Research Institute on Innovation and Technology, University of Limburg, Maastricht, Netherlands

108. Etzerodt, P., Madsen, K.H. 1988. Information Systems Assessment as a Learning Process, Proceedings of the IFIP WG 8.2 Working Conference on Information Systems Assessment, , North-Holland, Amsterdam, pp 333-345.

109. Duvall, P. M. 2007. Continuous integration – Improving Software Quality and Reducing Risk Addison-Wesley

110. Budgen, D., Thomson, M. 2001. CASE Tool Evaluation: Experiences From An Empirical Study. The Journal of Systems and Software , 67 (2003), 55–75

111. Kemerer, C.F., 1992. How The Learning Curve Affects CASE Tool Adoption. IEEE Software 9 (3), 23-28.

112. Iivari, J., 1996. Why Are CASE Tools Not Used? Communications of the ACM 39 (10), 94-103.

113. Continuous Integration, 2006.
http://martinfowler.com/articles/continuousIntegration.html
Last accessed: January 22, 2010

114. ISO 2001 Software Engineering – Product Quality – Part 1: Quality model, ISO/IEC 9126-1: 2001

TRADEMARKS

QuickBuild is a trademark of PmEase.
Cruise is a trademark of ThoughtWorks.
BuildForge is a trademark of IBM.
AnthillPro is a trademark of Urbancode.
Automated Build Studio is a trademark of AutomatedQA.
Bamboo is a trademark of Atlassian Software Systems.
EasyCIS is a trademark of Vaclav Zahradnik.
ElectricCommander is a trademark of Electric Cloud.
FinalBuilder is a trademark of Vsoft Technologies.
InstallAce is a trademark of DigiAce.
OpenMake Meister is a trademark of OpenMake Software.
OpenMake Mojo is a trademark of OpenMake Software.
Parabuild is a trademark of Viewtier Systems.
Pulse is a trademark of Zutubi.
TeamCity is a trademark of JetBrains.
Team Foundation Server is a trademark of Microsoft.
AccuRev is a trademark of AccuRev Inc.
BitKeeper is a trademark of BitMover Inc.
ClearCase is a trademark of IBM Rational.
Synergy is a trademark of Telelogic (IBM).
Co-Op is a trademark of Reliable Software.
Perforce is a trademark of Perforce Software Inc.
PureCM is a trademark of PureCM Ltd.
SourceAnywhere is a trademark of Dynamsoft Corporation.
Surround SCM is a trademark of Seapine Software.
Team Foundation Server is a trademark of Microsoft.
Vault is a trademark of SourceGear LLC.
VSS is a trademark of Microsoft.
Xcode is a trademark of Apple Inc.

APPENDICES

APPENDIX A:

LOW LEVEL CRITERION DEFINITIONS FOR EACH TOOL AREA

1. CI Server Tool Area

Table 41: LLC1

Item name	Description
ID	**LLC1**
Title	Feedback through email and RSS
Definition	The CI server shall support build result notifications. The notifications shall include but not limited to email and RSS. The message shall point to the build report in case of a successful build. The message shall point to the source that caused the error in case of a failed build. The CI server shall support filtering the mail recipients according to the build results.
Rationale	One of the main purposes of the continuous integration practice is the build result notification. By using notifications, all the people that are involved in developing the project can observe the health of it and can intervene quickly in case of build failures. However, to avoid big mail stacks, it is requested by the developers that the CI server should send email only in case of build failures, not after successful builds.
Source	Deployer, developers
Adaptability	Email notification functionality is of prime importance and it should be included in the tool. However the RSS notification functionality may be provided by an external plugin if it is not included in the system by default.
Priority	High
Type	Functional

Table 42: LLC2

Item name	Description
ID	**LLC2**
Title	Extension mechanisms
Definition	The CI server shall support being extended through plugins or other mechanisms.
Rationale	The functionality of the server should not be closed to modification. It should be extendable through plugins.
Source	Deployer
Adaptability	None
Priority	Normal
Type	Functional

Reliability requirement was detailed in criterion LLC3 and another functional requirement was derived from it as LLC4.

Table 43: LLC3

Item name	Description
ID	**LLC3**
Title	Robust working
Definition	The CI server shall execute its processes correctly and repeatedly once they are constructed.
Rationale	It is requested that the CI server be robust enough to require only minimal user intervention.
Source	Deployer
Adaptability	None
Priority	Normal
Type	Quality

Table 44: LLC4

Item name	Description
ID	**LLC4**
Title	Error pointing
Definition	The CI server shall be able to point the version of the file that is causing a build failure.
Rationale	The CI server should make it easy to pinpoint error sources.
Source	Deployer and developers
Adaptability	None
Priority	High
Type	Functional

The requirement named "longevity prospects" was expanded to LLC5 and LLC6.

Table 45: LLC5

Item name	Description
ID	**LLC5**
Title	Vendor or community stability
Definition	The vendor shall be in the sector for at least a year. If the tool is open source it shall have a large community.
Rationale	If the tool's vendor goes bankrupt, it would mean support discontinuance for the tool. In the case of open source software, a small user group would also mean insufficient support.
Source	Management, deployer
Adaptability	None
Priority	High
Type	Supplier/Community

Table 46: LLC6

Item name	Description
ID	**LLC6**
Title	Vendor and tool reputation
Definition	The vendor's credibility and its tool's success in the CI server area shall be high. In case of open source software, the original developer firm of the tool may be considered in this respect.
Rationale	High reputation of the vendor generally indicates high reliability.
Source	Management
Adaptability	None
Priority	Normal
Type	Supplier/Community

Table 47: LLC7

Item name	Description
ID	**LLC7**
Title	Weblogic support
Definition	If the server is distributed as a web application archive (WAR) file, it shall support Weblogic server version 9.2
Rationale	The department is using Weblogic application server so the applications that will run on it, should support it.
Source	Deployer
Adaptability	None
Priority	High
Type	Functional

Table 48: LLC8

Item name	Description
ID	**LLC8**
Title	Ease of use
Definition	The CI server shall include a web interface for configuration. The CI server shall support configuration via XML files.
Rationale	For easy maintenance of the tool, a web interface is requested. From that interface, full functionality of the tool should be configurable. Also, because there is Ant experience in the department, configurability via XML files is requested secondarily.
Source	Deployer
Adaptability	It is acceptable if the web frontend of the tool is provided from third-party firms.
Priority	High
Type	Functional

Table 49: LLC9

Item name	Description
ID	**LLC9**
Title	Ease of installation
Definition	The CI server shall be easily installable without requiring extensive configuration.
Rationale	It would be required in the future that the tool is reinstalled on a different host. This is not a rare case in the department.
Source	System administrators, deployer
Adaptability	The CI server may require post install configuration but if wizards are provided for this, then the CI server may pass from this criterion.
Priority	Normal
Type	Quality

Table 50: LLC10

Item name	Description
ID	**LLC10**
Title	Eclipse integration
Definition	The CI server shall support integration with Eclipse 3.4.2
Rationale	It is requested by the deployer that it should be possible to manage the CI server operations from his Eclipse IDE.
Source	Deployer
Adaptability	The integration may not be internal; instead it would be available by a third party plugin.
Priority	Normal
Type	Functional

Table 51: LLC11

Item name	Description
ID	**LLC11**
Title	Dashboard presence
Definition	The CI server shall include a web based dashboard that displays its most recent and former build details and results.
Rationale	It is requested that the operations of the build tool be examined easily from a web interface.
Source	Literature, deployer
Adaptability	None
Priority	High
Type	Functional

Table 52: LLC12

Item name	Description
ID	**LLC12**
Title	Labeling
Definition	The CI server shall support labeling its builds and shall also support modification of the labeling format.
Rationale	If the builds are labeled, referring to them becomes easier.
Source	Literature
Adaptability	None
Priority	High
Type	Functional

Table 53: LLC13

Item name	Description
ID	**LLC13**
Title	Project dependency support
Definition	The CI server shall support project dependencies so that when a build starts for a project, building of the dependent projects can be triggered.
Rationale	Some development projects which are ongoing in the department share libraries. Therefore a build performed on one of these projects should trigger the build on others.
Source	Literature, deployer
Adaptability	None
Priority	High
Type	Functional

Detailed bill of materials report requirement was detailed into two criteria: LLC14 and LLC15.

Table 54: LLC14

Item name	Description
ID	**LLC14**
Title	Bill of materials support
Definition	The CI server shall support reporting each build contents in a bill of materials report.
Rationale	For compliance requirements in the department, all of the artifacts included in a specific build should be able to be identified and listed.
Source	Literature, Management
Adaptability	The reporting format may be arbitrary however it should provide the necessary information.
Priority	High
Type	Functional

Table 55: LLC15

Item name	Description
ID	**LLC15**
Title	File fingerprinting support
Definition	The CI server shall be able to present the information about JAR version-build relationship.
Rationale	The CI server should keep track of which build produced which jar and which build is using which version of a jar. Interdependent modules are being developed by different developers who exchange JAR files in between. So keeping track of the trail of those JARs in builds is beneficial.
Source	Literature
Adaptability	None
Priority	Normal
Type	Functional

Table 56: LLC16

Item name	Description
ID	**LLC16**
Title	Role-based user management
Definition	The CI server shall be able to categorize users in roles.
Rationale	It is requested to define user permissions in groups representing roles such as the deployers or developers groups.
Source	Literature
Adaptability	None
Priority	Normal
Type	Functional

Table 57: LLC17

Item name	Description
ID	**LLC17**
Title	LDAP authentication
Definition	The CI server shall support Active Directory authentication.
Rationale	It is requested by the system administrators that the tool's authentication mechanisms support LDAP so that password management can be performed centrally.
Source	Literature, system administrators
Adaptability	None
Priority	Normal
Type	Functional

Table 58: LLC18

Item name	Description
ID	**LLC18**
Title	SCM filtering
Definition	The CI server shall be able to filter the file types that can trigger a build.
Rationale	The development platforms being used in the department includes many configuration and documentation files. Changes in those files do not affect the codebase and should not trigger the build operation.
Source	Literature, developers
Adaptability	None
Priority	High
Type	Functional

Table 59: LLC19

Item name	Description
ID	**LLC19**
Title	Multiple SCM repository support
Definition	The CI server shall be able to monitor multiple source repositories.
Rationale	The department is using multiple source repositories belonging to a single project. This structure is valid for the current SCM system however it will probably be the same in the to be acquired system.
Source	Literature
Adaptability	It is acceptable if this functionality is provided by a third party plugin.
Priority	Normal
Type	Functional

Table 60: LLC20

Item name	Description
ID	**LLC20**
Title	Graphical build trends
Definition	The CI server shall be able to present its build results over time on a graphical format that will be available on web.
Rationale	It is requested by the management that the trend of the builds should be observable on graphics which will be a quick way of project health determination.
Source	Literature, management
Adaptability	None
Priority	Normal
Type	Functional

2. Configuration Management Tool Area

Table 61: LLC21

Item name	Description
ID	**LLC21**
Title	Remote access
Definition	The configuration management tool shall support remote access to its source repositories through a fat client.
Rationale	Because of the security concerns in the department, it is not desired to keep the source code on developer's laptops. Web interfaces of the tools is not a solution for this case since developers do not want to make their daily work on web which would be slow and not integrate to their IDES.
Source	Management
Adaptability	None
Priority	High
Type	Functional

Table 62: LLC22

Item name	Description
ID	**LLC22**
Title	Folder level security
Definition	The configuration management tool shall support definition of user permissions at the folder level.
Rationale	Giving permissions at the repository level is not efficient and leads to creation of multiple unnecessary repositories. It is required that the permissions be able to be defined at the folder level.
Source	Deployer and developers
Adaptability	None
Priority	High
Type	Functional

Table 63: LLC23

Item name	Description
ID	**LLC23**
Title	Extensive Documentation
Definition	The configuration management tool shall include comprehensive documentation that details its each function.
Rationale	The tool documentation is the first resource to look in case of a problem or a need to learn a function of the tool.
Source	Developers
Adaptability	None
Priority	Normal
Type	Quality

Complete command set and GUI requirement was detailed into two criteria: LLC24, LLC25.

Table 64: LLC24

Item name	**Description**
ID	**LLC24**
Title	Extensive command set
Definition	The configuration management tool shall support invocation of all its major functionality through command line.
Rationale	Some developers are familiar with command line usage of tool and in fact they prefer such usage over GUIs. Also, sometimes it is required to write scripts that invoke command line.
Source	Developers
Adaptability	None
Priority	Normal
Type	Quality

Table 65: LLC25

Item name	Description
ID	**LLC25**
Title	Extensive graphical interfaces
Definition	The configuration management tool shall include GUIs for its major functions.
Rationale	Some developers prefer usage from the GUI.
Source	Developers
Adaptability	The tool may include just one interface and all of its functionality may be invoked from that interface without the need of another one. The tool may have an interface that is being provided as a plugin.
Priority	Normal
Type	Quality

Table 66: LLC26

Item name	Description
ID	**LLC26**
Title	Robustness
Definition	The configuration management tool shall be able to work flawlessly except in case of failure of its dependent systems like the network system.
Rationale	The configuration management tool holds the primary asset of the department: the code files. Therefore a failure may lead to data loss and significant rework effort.
Source	Developers
Adaptability	None
Priority	Normal
Type	Quality

Table 67: LLC27

Item name	Description
ID	**LLC27**
Title	Branching abilities
Definition	The configuration management tool shall support creation of branches at any level.
Rationale	The department is working on a new project and maintaining an older one. The development of the older one is still being done on a per request basis. Therefore branching abilities of the tool is important to diversify the work.
Source	Developers
Adaptability	None
Priority	Normal
Type	Functional

Table 68: LLC28

Item name	Description
ID	**LLC28**
Title	Eclipse integration
Definition	The configuration management tool shall support integration with Eclipse version 3.4.2. All of the major functions of the tool shall be able to be invoked inside Eclipse.
Rationale	This version of the Eclipse development platform is being used in the department. It is requested that all the tools in the department have integrations for Eclipse so that the developers would not need to switch programs and be able to do everything in Eclipse.
Source	Management, developers
Adaptability	None
Priority	High
Type	Functional

Table 69: LLC29

Item name	Description
ID	**LLC29**
Title	Line wise history tracking
Definition	The configuration management tool shall be able to show the originating version of each line in a text based code file.
Rationale	It is requested by the developers that the system should be able to show the line wise history in a java file. That is in when each line is added and by whom.
Source	Literature
Adaptability	None
Priority	Normal
Type	Functional

Table 70: LLC30

Item name	Description
ID	**LLC30**
Title	Uncommitted data indication
Definition	The configuration management tool shall be able to show the work that is not yet committed to the source repository.
Rationale	It is requested by the developers that the tool should list their files that are checked out but not yet committed.
Source	Literature, developers
Adaptability	None
Priority	Normal
Type	Functional

Table 71: LLC31

Item name	Description
ID	**LLC31**
Title	Per-file commit messages
Definition	The configuration management shall support comment entering for each commit.
Rationale	If the developers can enter a comment that accompanies a commit, they can quickly obtain data about each version creation provided that the comments are meaningful.
Source	Literature
Adaptability	None
Priority	Normal
Type	Functional

Table 72: LLC32

Item name	Description
ID	**LLC32**
Title	Binary file handling
Definition	The configuration management tool shall be able to version binary files as well as text files.
Rationale	The class files are not version controlled since they are reproducible however there are some types of binary configuration files that the developers need to version.
Source	Literature
Adaptability	None
Priority	High
Type	Functional

Table 73: LLC33

Item name	Description
ID	**LLC33**
Title	Reporting options
Definition	The configuration management tool shall include reporting facilities.
Rationale	It is requested by the management that the configuration management tool provide them metrics such as line of code measurement.
Source	Literature, management
Adaptability	Reporting options may be provided by external plugins.
Priority	Normal
Type	Functional

Optimistic locking requirement was detailed into two criteria: LLC34 and LLC35.

Table 74: LLC34

Item name	Description
ID	**LLC34**
Title	Optimistic locking support
Definition	The configuration management tool shall support the copy-modify-merge model for concurrent development.
Rationale	There are long configuration files which should be worked on by different developers. This model enables the developers to work in parallel on such files without to need to wait each other.
Source	Literature, developers
Adaptability	None
Priority	High
Type	Functional

Table 75: LLC35

Item name	Description
ID	**LLC35**
Title	Merging support
Definition	The configuration management tool shall support merging that is reconciling multiple changes performed on different copies of the same file.
Rationale	In the optimistic locking model merging is necessary.
Source	Literature, developers
Adaptability	None
Priority	High
Type	Functional

Table 76: LLC36

Item name	Description
ID	**LLC36**
Title	Labeling support
Definition	The configuration management tool shall support branch and version labeling.
Rationale	Labeling is required to identify baselines (milestones). Also, the CI server typically designates a version that is used in the build by giving a label to it.
Source	Deployer
Adaptability	None
Priority	High
Type	Functional

Table 77: LLC37

Item name	Description
ID	**LLC37**
Title	Embedded database
Definition	The configuration management tool shall include an embedded database suited for file handling.
Rationale	It is not desired to use a RDBMS to keep the source because these types of databases are not designed for this purpose. Also, it is not desired to deal with the extra configuration of a database. It would be better if the tool includes a preconfigured embedded database suited for file storage.
Source	Deployer, Management
Adaptability	None
Priority	Normal
Type	Functional

3. Build Tool Area

Table 78: LLC38

Item name	Description
ID	**LLC38**
Title	XML Syntax
Definition	The build tool shall support XML in the definition of build scripts.
Rationale	All the developers use XML in the development. They are familiar to working in XML.
Source	Deployer, developers
Adaptability	None
Priority	High
Type	Functional

Table 79: LLC39

Item name	Description
ID	**LLC39**
Title	Easy installation
Definition	The build tool shall be able to be installed easily by an installer.
Rationale	Hardware and operating system changes occur in regular intervals. The system administrators prefer easily installable tools.
Source	System administrators
Adaptability	None
Priority	Normal
Type	Quality

Table 80: LLC40

Item name	Description
ID	**LLC40**
Title	Easy project configuration
Definition	The build tool shall support project creation in a few steps.
Rationale	The creation and configuration of a project in Ant is straightforward and users request the same behavior from the to be acquired build tool.
Source	Deployer, developers
Adaptability	None
Priority	Normal
Type	Quality

The requirement named "Time to learn for a new developer" was detailed into two criteria LLC40 and LLC41.

Table 81: LLC41

Item name	Description
ID	**LLC41**
Title	Shallow learning curve
Definition	The build tool shall include low complexity functions.
Rationale	Dealing with the complexities of a build tool is not preferred since the reason of its presence is easing the build process.
Source	Deployer, developers
Adaptability	None
Priority	Normal
Type	Quality

Table 82: LLC42

Item name	Description
ID	**LLC42**
Title	Complete documentation
Definition	The build tool shall include documentation for all of its functionality.
Rationale	Information is requested to be easily found for the build tool since the build schemes change regularly and capabilities of the tool which are not needed before may be needed.
Source	Literature, developers
Adaptability	None
Priority	Normal
Type	Quality

Table 83: LLC43

Item name	Description
ID	**LLC43**
Title	Multi-project support
Definition	The build tool shall support defining and executing multiple projects.
Rationale	The structure of the codebase necessitates multiple build projects. That is the current condition in Ant.
Source	Deployer, developers
Adaptability	Another functionality that results in the same behavior as multiple projects may be accepted.
Priority	High
Type	Functional

Table 84: LLC44

Item name	Description
ID	**LLC44**
Title	Log generation
Definition	The build tool shall support generation of log files belonging to builds executed. These logs shall include each action executes by the tool and its result.
Rationale	Logging of all actions is requested by the standards being conformed in the department.
Source	Deployer, developers
Adaptability	None
Priority	High
Type	Functional

Table 85: LLC45

Item name	Description
ID	**LLC45**
Title	Eclipse integration
Definition	The build shall support integration with Eclipse version 3.4.2
Rationale	Developers request to run their build scripts inside their Eclipse shells. They don't want to switch platforms.
Source	Literature, developers
Adaptability	None
Priority	High
Type	Functional

APPENDIX B:

EVALUATION RESULTS FOR EACH LOW LEVEL CRITERION

1. Assessment Details, Findings and Comparisons for CI server tool area:

Evaluation and comparisons for the following tools in CI server tool area are given in Tables 86-95.

Tools Evaluated: CruiseControl (Version: 2.8.2)
Hudson (Version: 1.341)
QuickBuild (Version: 2.0.15)
TeamCity (Version: 5.0)

Table 86: Evaluation and comparison table for LLC2

Criterion ID	**LLC2**			
Criterion Name	**Extension mechanisms**			
CruiseControl	CC is open source so it is extendable directly from source. It also has a plugin development resource page and many developed plugins [16].			
Hudson	Hudson is open source and it has a plugin base which is not big as CruiseControl but constantly growing [17].			
QuickBuild	QuickBuild is not open source. It has plugins available however they are mostly developed by its vendor PMEase Inc. and the evaluators couldn't find any information about developing custom plugins. Therefore it is concluded that users are dependent on the vendor for extension [18].			
TeamCity	TeamCity is not open source however it has a sufficient plugin development facility and documentation for extension [19].			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	3	5	3
Hudson	1/3	1	5	1
QuickBuild	1/5	1/5	1	1/5
TeamCity	1/3	1	5	1

Table 87: Evaluation and comparison table for LLC3

Criterion ID	**LLC3**			
Criterion Name	**Robust working**			
CruiseControl	CruiseControl is in the market since 2002 and it has 27 versions released since then. This is an indication of high stability of the product. A company using it for years also gave good feedback about the tool's robustness.			
Hudson	Hudson is relatively new in the market. It has been available since 2007. It has an issue tracker reference at its page where the details about the product's defects can be found. We can conclude from this reference that the tool has a good rate of defect fixing which is a good indication of its robustness.			
QuickBuild	PMEase, the developer firm of QuickBuild, has been in the market for a considerable amount of time. They also developed the predecessor; LuntBuild: an open source tool containing functionality close to QuickBuild. The tool has a defect reference page where we can see a high defect fixing rate [20].			
TeamCity	TeamCity is in the market since 2006. The tool has a defect reference page where we can see that it has a relatively high number of open defects (2622 defects) which is a negative indicator of robustness.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	2	2	5
Hudson	1/2	1	1	3
QuickBuild	1/2	1	1	3
TeamCity	1/5	1/3	1/3	1

Table 88: Evaluation and comparison table for LLC6

Criterion ID	**LLC6**			
Criterion Name	**Vendor and tool reputation**			
CruiseControl	CruiseControl is open to public however it is originally developed by ThoughtWorks which is a well known company in the sector with many references. Also the tool itself has a very high download count in sourceforge.			
Hudson	Hudson is also open to public. Its producer is Kohsuke Kawaguchi, a developer working for Sun Microsystems. Commercial support for Hudson is also announced by Sun in 2009. The tool has earned an award in developer solutions category in 2008 in the JavaOne conference [21].			
QuickBuild	QuickBuild has many references that can be seen in its site however its name is not mentioned in reports or articles discussing the continuous integration market. Its producer firm PMEase Inc. does not have reputation as much as the other candidate tools that the evaluators selected.			
TeamCity	TeamCity has gained relatively high interest than the other tools in the commercial CI server tool market according to the articles and company references. However, continuous integration is not the main focus of its producer firm; JetBrains. Instead they mostly focus on their JAVA IDE: IntelliJ IDEA. This raises concerns about the future of the tool.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	2	6	3
Hudson	1/2	1	3	2
QuickBuild	1/6	1/3	1	1/4
TeamCity	1/3	1/2	4	1

Table 89: Evaluation and comparison table for LLC9

Criterion ID	**LLC9**			
Criterion Name	**Ease of installation**			
CruiseControl	Cruise Control contains both an executable binary package and the source code version. According to the evaluators' trial on the real tool, it is very easy to make an installation with the executable distribution. Minimal configuration is necessary to make the tool running since it comes in a self contained package including a Jetty server.			
Hudson	Hudson also contains a binary and a source distribution. The war file that comes can be directly executed with java- jar or it can be deployed to a J2EE compliant application server. If it is directly executed it will be served from the Winstone servlet container that is coming bundled with it. So installation can be done in a quick and straightforward way.			
QuickBuild	Setup of QuickBuild is not difficult however if a multiplatform parallel system will be configured it requires agent installations and some configuration. After starting its server, the web interface launches a wizard for further configuration. The package is self contained in that it contains both an application server and a database.			
TeamCity	TeamCity has an easy to follow step by step documentation for installation. Therefore, its installation can be considered quite easy. As in the case of QuickBuild configuration of build agents can require a little more effort.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	1	2	2
Hudson	1	1	2	2
QuickBuild	1/2	1/2	1	1
TeamCity	1/2	1/2	1	1

Table 90: Evaluation and comparison table for LLC10

Criterion ID	**LLC10**			
Criterion Name	**Eclipse integration**			
CruiseControl	Eclipse 3.2 and higher are supported with the latest Eclipse plugin of the tool [22].			
Hudson	Eclipse 3.4.2 is supported by the Hudson's Eclipse plugin [23].			
QuickBuild	The support for Eclipse is declared in the QuickBuild's site but details about the version of Eclipse that is supported or an installation procedure could not be found. This condition raises concerns about the quality of the integration.			
TeamCity	Eclipse 3.4 is supported by TeamCity by installing the plugin that comes with the product. Installation is described step by step and is quite easy.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	1	4	1
Hudson	1	1	4	1
QuickBuild	1/4	1/4	1	1/4
TeamCity	1	1	4	1

Table 91: Evaluation and comparison table for LLC15

Criterion ID	**LLC15**			
Criterion Name	**File fingerprinting support**			
CruiseControl	This or a similar functionality could not be found in CruiseControl. It will be considered as non-existent.			
Hudson	Hudson directly supports file fingerprinting as detailed in its documentation. In case of multiple dependent projects, Hudson can easily present which artifact of one project is used in the other project. This can be achieved at the JAR level.			
QuickBuild	According to the information obtained directly from the vendor that QuickBuild does not provide JAR level fingerprinting but includes a detailed dependency tracking facility that can provide that information indirectly.			
TeamCity	TeamCity also has an equivalent dependency tracking system for artifacts that can show which artifact is used in which project but this functionality is not for JAR level tracking.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	1/9	1/6	1/6
Hudson	9	1	4	5
QuickBuild	6	1/4	1	2
TeamCity	6	1/5	1/2	1

Table 92: Evaluation and comparison table for LLC16

Criterion ID	**LLC16**			
Criterion Name	**Role-based user management**			
CruiseControl	CruiseControl has user authentication but it is per user based and role or scheme definitions are not supported. Also, available permission definitions are very limited.			
Hudson	Hudson has group based authentication. Thus role groups can be defined with different permission levels.			
QuickBuild	QuickBuild possesses both user and group based authentication mechanisms and it offers the ability to setup permissions at a finer level. A role-based model can be easily constructed.			
TeamCity	TeamCity offers a role-based authentication mechanism by default. The tool comes with the predefined system administrator, project administrator, project developer, agent manager and project viewer roles. New roles can be added, permissions are configurable.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	1/5	1/7	1/8
Hudson	5	1	1/2	1/3
QuickBuild	7	2	1	1/2
TeamCity	8	3	2	1

Table 93: Evaluation and comparison table for LLC17

Criterion ID	**LLC17**			
Criterion Name	**LDAP authentication**			
CruiseControl	CruiseControl does not support LDAP authentication.			
Hudson	LDAP authentication is supported by Hudson.			
QuickBuild	LDAP authentication is supported by QuickBuild.			
TeamCity	LDAP authentication is supported by TeamCity however it should be configured via an XML file not from the GUI which makes the configuration a little more difficult. Though the tool also has an LDAP synchronization functionality which can fetch user and group data from LDAP and update user group memberships based on the data retrieved.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	1/9	1/9	1/9
Hudson	9	1	1	1
QuickBuild	9	1	1	1
TeamCity	9	1	1	1

Table 94: Evaluation and comparison table for LLC19

Criterion ID	**LLC19**			
Criterion Name	**Multiple SCM repository support**			
CruiseControl	CruiseControl can poll multiple SCM systems and multiple repositories.			
Hudson	Hudson cannot be configured to connect with multiple SCM systems however it can be configured to poll multiple SCM repositories of the same SCM system.			
QuickBuild	QuickBuild can poll multiple SCM systems and multiple repositories.			
TeamCity	TeamCity can poll multiple SCM systems and multiple repositories.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	4	1	1
Hudson	1/4	1	1/4	1/4
QuickBuild	1	4	1	1
TeamCity	1	4	1	1

Table 95: Evaluation and comparison table for LLC20

Criterion ID	**LLC20**			
Criterion Name	**Graphical build trends**			
CruiseControl	The Cruise Control's dashboard interface provides visualization of build results. The department couldn't find any graphical build trend representation in standalone CruiseControl however this functionality is available through various plugins. Configuration of dashboard should be done via an XML file.			
Hudson	Hudson also includes a dashboard interface in which several graphical representations are offered. However trend graphics are mainly for test tool integrations and test results. Build result graphics are comparably limited.			
QuickBuild	QuickBuild includes statistical graphs that can show build and test trends.			
TeamCity	TeamCity offers custom graphic creation functionality which allows the user to create a graphical representation of the build data. This is a strong point in that the user can create any type of chart however the configuration should be made via an XML file.			
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**
CruiseControl	1	1/3	1/7	1/6
Hudson	3	1	1/6	1/4
QuickBuild	7	6	1	2
TeamCity	6	4	1/2	1

2. **Assessment Details, Findings and Comparisons for Configuration Management Tool Area:**

Only tool that passed the screening activity was ClearCase. Therefore, evaluation and comparison efforts were not necessary for the configuration management tool area. However, for completeness and future reference assessments were also made for ClearCase. Results are given in Tables 96-105.

Tool Evaluated: ClearCase (Version: 7.1.1)

Table 96: Evaluation and comparison table for LLC23

Criterion ID	**LLC23**
Criterion Name	**Extensive documentation**
ClearCase	ClearCase is coming with extensive documentation detailing its every feature. Guides are available and separated as administrator guide, user guide and installation guide. Users can get help inside the tool by clicking on help in the top bar. Also context sensitive help that is accessible by clicking on F1 is available for most of the windows that the tool has.

Table 97: Evaluation and comparison table for LLC24

Criterion ID	**LLC24**
Criterion Name	**Extensive command set**
ClearCase	All of the ClearCase functionality is available in the tool's command set since the tool is firstly developed for Unix and then ported on Windows.

Table 98: Evaluation and comparison table for LLC25

Criterion ID	**LLC25**
Criterion Name	**Extensive graphical interfaces**
ClearCase	Most major operations of the tool are included in its GUI called ClearCase Explorer. Wizards are generally offered for multi-step tasks. Also, a graphical interface is available for the administrators of the tool called "administration console".

Table 99: Evaluation and comparison table for LLC26

Criterion ID	**LLC26**
Criterion Name	**Robustness**
ClearCase	ClearCase entered the CASE tool market in 1992 for Unix platforms. So it is in the market for 17 years and many versions are released during this period. With every version many defects are fixed however addition of new functionalities also brought new defects and increased the tool complexity. According to the research, the tool can be considered robust for its core functionalities however fragile for human errors due to its complexity. So, errors are likely to be faced in the learning and adjustment period. The defect fixing activities are being conducted by IBM regularly.

Table 100: Evaluation and comparison table for LLC27

Criterion ID	**LLC27**
Criterion Name	**Branching abilities**
ClearCase	ClearCase offers many facilities for branching. The usage model is divided into two as base usage and UCM (unified change management) usage models. In the base model, branching is manual and there is no limit for branch creation. These branches can also be merged. The UCM model automates the branch formation and merging operations. Either way, branching needs for all SCM patterns can be satisfied.

Table 101: Evaluation and comparison table for LLC29

Criterion ID	**LLC29**
Criterion Name	**Line wise history tracking**
ClearCase	ClearCase offers the "cleartool annotate" command in its command-set which takes a text file as input and displays it with information added after each line. This information indicates when, and in which version, the line was added. So we can conclude that this criterion is fully satisfied by ClearCase.

Table 102: Evaluation and comparison table for LLC30

Criterion ID	**LLC30**
Criterion Name	**Uncommitted data indication**
ClearCase	Through the usage of ClearCase's "find checkouts" functionality, the files that are checked out but not yet committed back to the repository can easily be found. This functionality can be invoked from the command set, GUI or from supported IDE plugins.

Table 103: Evaluation and comparison table for LLC31

Criterion ID	**LLC31**
Criterion Name	**Per-file commit messages**
ClearCase	ClearCase supports entering comments for all the commit operations. Usage of this capability is optional but can also be made mandatory by the tool.

Table 104: Evaluation and comparison table for LLC33

Criterion ID	**LLC33**
Criterion Name	**Reporting options**
ClearCase	ClearCase includes an interface called "Report Builder" which consists of various predefined reports that are centered on artifacts and users. Custom reports can also be added but they should be prepared in the proper format required by the tool.

Table 105: Evaluation and comparison table for LLC37

Criterion ID	**LLC37**
Criterion Name	**Embedded database**
ClearCase	ClearCase is bundled with an embedded database and does not require or support a third party database. The database coming with the tool has a specialized and secured repository format for file storage and retrieval. It only keeps the whole file once and then starts to keep only the differences between its versions. This avoids excessive disk space usage but is only valid for text based files.

3. Assessment Details, Findings and Comparisons for Build Tool Area:

Evaluations and comparisons for the following tools in Build Tool Area are given in Tables 106-109.

Tools Evaluated: Ant (Version: 1.8.0RC1)
Maven (Version: 2.2.1)

Table 106: Evaluation and comparison table for LLC379

Criterion ID	**LLC39**	
Criterion Name	**Easy installation**	
Ant	Ant may come bundled with other applications like an IDE. Otherwise, it can be installed very easily. After expanding the installation package, some environment variables should be set in order Ant to work correctly. The installation and environment variable configuration may not exceed 5 minutes.	
Maven	Maven can also be installed easily. It only requires a standard JDK on the computer.	
Comparison	**Ant**	**Maven**
Ant	1	1
Maven	1	1

Table 107: Evaluation and comparison table for LLC40

Criterion ID	**LLC40**	
Criterion Name	**Easy project configuration**	
Ant	Project configuration with Ant can get detailed for complex builds. However, it allows extensive customization of the build scripts.	
Maven	Maven offers a standard project layout and if the user conforms to this layout, project configuration is quite easy and straightforward.	
Comparison	**Ant**	**Maven**
Ant	1	1/4
Maven	4	1

Table 108: Evaluation and comparison table for LLC41

Criterion ID	**LLC41**	
Criterion Name	**Shallow learning curve**	
Ant	Since Ant is currently being used in the department, everybody knows its syntax and functions. However, it is also stated by the developers that learning Ant for a newcomer would not be hard.	
Maven	Maven has a relatively different project structure and it includes dependency management facilities which are not familiar to the department. However, according to the users in the community, it is not difficult to learn.	
Comparison	**Ant**	**Maven**
Ant	1	4
Maven	1/4	1

Table 109: Evaluation and comparison table for LLC42

Criterion ID	**LLC42**	
Criterion Name	**Complete documentation**	
Ant	Ant comes with a good documentation which covers each functionality of the tool and is continually updated by Apache.	
Maven	Maven also has an extensive and up-to-date documentation at Apache's site.	
Comparison	**Ant**	**Maven**
Ant	1	1
Maven	1	1

APPENDIX C: PRORITY CALCULATIONS OF ALTERNATIVES

Calculations for Activity 6 Step 1.4 "Derivation of ratio scale priorities for the alternatives" are included in this section for both the CI server and build tool areas.

Synthesis tables for the CI server tool area:

Table 110: Synthesis table for LLC2

Criterion ID	**LLC2**				
Criterion Name	**Extension mechanisms**				
Comparison	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**	
CruiseControl	1	3	5	3	
Hudson	1/3	1	5	1	
QuickBuild	1/5	1/5	1	1/5	
TeamCity	1/3	1	5	1	
Total	1.867	5.200	16.000	5.200	
Synthesis	**CruiseControl**	**Hudson**	**QuickBuild**	**TeamCity**	**Row Average**
CruiseControl	0.536	0.577	0.313	0.577	0.501
Hudson	0.178	0.192	0.313	0.192	0.219
QuickBuild	0.107	0.038	0.063	0.038	0.062
TeamCity	0.178	0.192	0.313	0.192	0.219
Total					1.000

Table 111: Synthesis table for LLC3

Criterion ID	LLC3				
Criterion Name	Robust working				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	2	2	5	
Hudson	1/2	1	1	3	
QuickBuild	1/2	1	1	3	
TeamCity	1/5	1/3	1/3	1	
Total	2.200	4.333	4.333	12.000	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.455	0.462	0.462	0.417	0.449
Hudson	0.227	0.231	0.231	0.250	0.235
QuickBuild	0.227	0.231	0.231	0.250	0.235
TeamCity	0.091	0.077	0.077	0.083	0.082
Total					1.000

Table 112: Synthesis table for LLC6

Criterion ID	LLC6				
Criterion Name	Vendor and tool reputation				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	2	6	3	
Hudson	1/2	1	3	2	
QuickBuild	1/6	1/3	1	1/4	
TeamCity	1/3	1/2	4	1	
Total	2.000	3.833	14.000	6.250	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.500	0.522	0.429	0.480	0.483
Hudson	0.250	0.261	0.214	0.320	0.261
QuickBuild	0.083	0.087	0.071	0.040	0.070
TeamCity	0.167	0.130	0.286	0.160	0.186
Total					1.000

Table 113: Synthesis table for LLC9

Criterion ID	LLC9				
Criterion Name	Ease of installation				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	1	2	2	
Hudson	1	1	2	2	
QuickBuild	1/2	1/2	1	1	
TeamCity	1/2	1/2	1	1	
Total	3.000	3.000	6.000	6.000	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.333	0.333	0.333	0.333	0.333
Hudson	0.333	0.333	0.333	0.333	0.333
QuickBuild	0.167	0.167	0.167	0.167	0.167
TeamCity	0.167	0.167	0.167	0.167	0.167
Total					1.000

Table 114: Synthesis table for LLC10

Criterion ID	LLC10				
Criterion Name	Eclipse integration				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	1	4	1	
Hudson	1	1	4	1	
QuickBuild	1/4	1/4	1	1/4	
TeamCity	1	1	4	1	
Total	3.250	3.250	13.000	3.250	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.308	0.308	0.308	0.308	0.308
Hudson	0.308	0.308	0.308	0.308	0.308
QuickBuild	0.077	0.077	0.077	0.077	0.077
TeamCity	0.308	0.308	0.308	0.308	0.308
Total					1.000

Table 115: Synthesis table for LLC15

Criterion ID	LLC15				
Criterion Name	File fingerprinting support				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	1/9	1/6	1/6	
Hudson	9	1	4	5	
QuickBuild	6	1/4	1	2	
TeamCity	6	1/5	1/2	1	
Total	22.000	1.561	5.667	8.167	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.045	0.071	0.029	0.020	0.042
Hudson	0.409	0.641	0.706	0.612	0.592
QuickBuild	0.273	0.160	0.176	0.245	0.214
TeamCity	0.273	0.128	0.088	0.122	0.153
Total					1.000

Table 116: Synthesis table for LLC16

Criterion ID	LLC16				
Criterion Name	Role-based user management				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	1/5	1/7	1/8	
Hudson	5	1	1/2	1/3	
QuickBuild	7	2	1	1/2	
TeamCity	8	3	2	1	
Total	21.000	6.200	3.643	1.958	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.048	0.032	0.039	0.064	0.046
Hudson	0.238	0.161	0.137	0.170	0.177
QuickBuild	0.333	0.323	0.274	0.255	0.296
TeamCity	0.381	0.484	0.549	0.511	0.481
Total					1.000

Table 117: Synthesis table for LLC17

Criterion ID	LLC17				
Criterion Name	LDAP authentication				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	1/9	1/9	1/9	
Hudson	9	1	1	1	
QuickBuild	9	1	1	1	
TeamCity	9	1	1	1	
Total	28.000	3.111	3.111	3.111	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.036	0.036	0.036	0.036	0.036
Hudson	0.321	0.321	0.321	0.321	0.321
QuickBuild	0.321	0.321	0.321	0.321	0.321
TeamCity	0.321	0.321	0.321	0.321	0.321
Total					1.000

Table 118: Synthesis table for LLC19

Criterion ID	LLC19				
Criterion Name	Multiple SCM repository support				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	4	1	1	
Hudson	1/4	1	1/4	1/4	
QuickBuild	1	4	1	1	
TeamCity	1	4	1	1	
Total	3.250	13.000	3.250	3.250	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.308	0.308	0.308	0.308	0.308
Hudson	0.077	0.077	0.077	0.077	0.077
QuickBuild	0.308	0.308	0.308	0.308	0.308
TeamCity	0.308	0.308	0.308	0.308	0.308
Total					1.000

Table 119: Synthesis table for LLC20

Criterion ID	LLC20				
Criterion Name	Graphical build trends				
Comparison	CruiseControl	Hudson	QuickBuild	TeamCity	
CruiseControl	1	1/3	1/7	1/6	
Hudson	3	1	1/6	1/4	
QuickBuild	7	6	1	2	
TeamCity	6	4	1/2	1	
Total	17.000	11.333	1.810	3.417	
Synthesis	CruiseControl	Hudson	QuickBuild	TeamCity	Row Average
CruiseControl	0.059	0.029	0.079	0.049	0.054
Hudson	0.176	0.088	0.092	0.073	0.107
QuickBuild	0.412	0.529	0.552	0.585	0.520
TeamCity	0.353	0.353	0.276	0.293	0.319
Total					1.000

Synthesis tables for the build tool area:

Table 120: Synthesis table for LLC39

Criterion ID	LLC39		
Criterion Name	Easy installation		
Comparison	Ant	Maven	
Ant	1	1	
Maven	1	1	
Total	2	2	
Synthesis	Ant	Maven	Row Average
Ant	0.500	0.500	0.500
Maven	0.500	0.500	0.500
Total			1.000

Table 121: Synthesis table for LLC40

Criterion ID	LLC40		
Criterion Name	Easy project configuration		
Comparison	Ant	Maven	
Ant	1	1/4	
Maven	4	1	
Total	5	1.25	
Synthesis	Ant	Maven	Row Average
Ant	0.200	0.200	0.200
Maven	0.800	0.800	0.800
Total			1.000

Table 122: Synthesis table for LLC41

Criterion ID	LLC41		
Criterion Name	Shallow learning curve		
Comparison	Ant	Maven	
Ant	1	4	
Maven	1/4	1	
Total	1.25	5	
Synthesis	Ant	Maven	Row Average
Ant	0.800	0.800	0.800
Maven	0.200	0.200	0.200
Total			1.000

Table 123: Synthesis table for LLC42

Criterion ID	LLC42		
Criterion Name	Complete documentation		
Comparison	Ant	Maven	
Ant	1	1	
Maven	1	1	
Total	2	2	
Synthesis	Ant	Maven	Row Average
Ant	0.500	0.500	0.500
Maven	0.500	0.500	0.500
Total			1.000

APPENDIX D: CONSISTENCY CALCULATIONS

Consistency check tables for the CI server tool area are given in this section.

Calculations for Activity 6 Step 1.5 - Consistency ratio estimation:

Table 124: Consistency check table for LLC2

Criterion ID	**LLC2**					
Criterion Name	**Extension mechanisms**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.501	3 X 0.219	5 X 0.062	3 X 0.219	2.249	4.489
Hudson	1/3 X 0.501	1 X 0.219	5 X 0.062	1 X 0.219	0.915	4.178
QuickBuild	1/5 X 0.501	1/5 X 0.219	1 X 0.062	1/5 X 0.219	0.221	3.567
TeamCity	1/3 X 0.501	1 X 0.219	5 X 0.062	1 X 0.219	0.915	4.178
Average						4.103
CI	(4.103 – 4) / 3 = 0.034					
CR	0.034 / 0.90 = 0.038					

Table 125: Consistency check table for LLC3

Criterion ID	**LLC3**					
Criterion Name	**Robust working**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.449	2 X 0.235	2 X 0.235	5 X 0.082	1.799	4.007
Hudson	1/2 X 0.449	1 X 0.235	1 X 0.235	3 X 0.082	0.941	4.002
QuickBuild	1/2 X 0.449	1 X 0.235	1 X 0.235	3 X 0.082	0.941	4.002
TeamCity	1/5 X 0.449	1/3 X 0.235	1/3 X 0.235	1 X 0.082	0.328	4.006
Average						4.004
CI	0.001					
CR	0.002					

Table 126: Consistency check table for LLC6

Criterion ID	**LLC6**					
Criterion Name	**Vendor and tool reputation**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.483	2 X 0.261	6 X 0.070	3 X 0.186	1.983	4.106
Hudson	1/2 X 0.483	1 X 0.261	3 X 0.070	2 X 0.186	1.085	4.155
QuickBuild	1/6 X 0.483	1/3 X 0.261	1 X 0.070	1/4 X 0.186	0.284	4.057
TeamCity	1/3 X 0.483	1/2 X 0.261	4 X 0.070	1 X 0.186	0.758	4.073
Average						4.098
CI	0.033					
CR	0.036					

Table 127: Consistency check table for LLC9

Criterion ID	**LLC9**					
Criterion Name	**Ease of installation**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.333	1 X 0.333	2 X 0.167	2 X 0.167	1.334	4.006
Hudson	1 X 0.333	1 X 0.333	2 X 0.167	2 X 0.167	1.334	4.006
QuickBuild	1/2 X 0.333	1/2 X 0.333	1 X 0.167	1 X 0.167	0.667	3.994
TeamCity	1/2 X 0.333	1/2 X 0.333	1 X 0.167	1 X 0.167	0.667	3.994
Average						4.000
CI	0.000					
CR	0.000					

Table 128: Consistency check table for LLC10

Criterion ID	**LLC10**					
Criterion Name	**Eclipse integration**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.308	1 X 0.308	4 X 0.077	1 X 0.308	1.232	4.000
Hudson	1 X 0.308	1 X 0.308	4 X 0.077	1 X 0.308	1.232	4.000
QuickBuild	1/4 X 0.308	1/4 X 0.308	1 X 0.077	1/4 X 0.308	0.308	4.000
TeamCity	1 X 0.308	1 X 0.308	4 X 0.077	1 X 0.308	1.232	4.000
Average						4.000
CI	0.000					
CR	0.000					

Table 129: Consistency check table for LLC15

Criterion ID	LLC15					
Criterion Name	File fingerprinting support					
Consistency	Cruise Control	Hudson	Quick Build	TeamCity	Totals	Division
CruiseControl	1 X 0.042	1/9 X 0.592	1/6 X 0.214	1/6 X 0.153	0.169	4.022
Hudson	9 X 0.042	1 X 0.592	4 X 0.214	5 X 0.153	2.591	4.377
QuickBuild	6 X 0.042	1/4 X 0.592	1 X 0.214	2 X 0.153	0.920	4.299
TeamCity	6 X 0.042	1/5 X 0.592	1/2 X 0.214	1 X 0.153	0.630	4.120
Average						4.205
CI	0.068					
CR	0.076					

Table 130: Consistency check table for LLC16

Criterion ID	LLC16					
Criterion Name	Role-based user management					
Consistency	Cruise Control	Hudson	Quick Build	TeamCity	Totals	Division
CruiseControl	1 X 0.046	1/5 X 0.177	1/7 X 0.296	1/8 X 0.481	0.184	3.996
Hudson	5 X 0.046	1 X 0.177	1/2 X 0.296	1/3 X 0.481	0.715	4.041
QuickBuild	7 X 0.046	2 X 0.177	1 X 0.296	1/2 X 0.481	1.213	4.096
TeamCity	8 X 0.046	3 X 0.177	2 X 0.296	1 X 0.481	1.972	4.100
Average						4.058
CI	0.019					
CR	0.022					

Table 131: Consistency check table for LLC17

Criterion ID	**LLC17**					
Criterion Name	**LDAP authentication**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.036	1/9 X 0.321	1/9 X 0.321	1/9 X 0.321	0.143	3.972
Hudson	9 X 0.036	1 X 0.321	1 X 0.321	1 X 0.321	1.287	4.009
QuickBuild	9 X 0.036	1 X 0.321	1 X 0.321	1 X 0.321	1.287	4.009
TeamCity	9 X 0.036	1 X 0.321	1 X 0.321	1 X 0.321	1.287	4.009
Average						4.000
CI	0.000					
CR	0.000					

Table 132: Consistency check table for LLC19

Criterion ID	**LLC19**					
Criterion Name	**Multiple SCM repository support**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.308	4 X 0.077	1 X 0.308	1 X 0.308	1.232	4.000
Hudson	1/4 X 0.308	1 X 0.077	1/4 X 0.308	1/4 X 0.308	0.308	4.000
QuickBuild	1 X 0.308	4 X 0.077	1 X 0.308	1 X 0.308	1.232	4.000
TeamCity	1 X 0.308	4 X 0.077	1 X 0.308	1 X 0.308	1.232	4.000
Average						4.000
CI	0.000					
CR	0.000					

Table 133: Consistency check table for LLC20

Criterion ID	**LLC20**					
Criterion Name	**Graphical build trends**					
Consistency	**Cruise Control**	**Hudson**	**Quick Build**	**TeamCity**	**Totals**	**Division**
CruiseControl	1 X 0.054	1/3 X 0.107	1/7 X 0.520	1/6 X 0.319	0.217	4.021
Hudson	3 X 0.054	1 X 0.107	1/6 X 0.520	1/4 X 0.319	0.435	4.069
QuickBuild	7 X 0.054	6 X 0.107	1 X 0.520	2 X 0.319	2.178	4.188
TeamCity	6 X 0.054	4 X 0.107	1/2 X 0.520	1 X 0.319	1.331	4.172
Average						4.113
CI	0.038					
CR	0.042					