Erasmus Universiteit Rotterdam

Bachelor Thesis Project

Traczilla: Dynamic Workflow Management

"Using Software Configuration Management Tools, to manage the dynamic behaviour of business processes."

Author:
Shawn Mclaren

Supervisor:
Emiel Caron

Co-reader:
Flavius Frasincar

Economie & Informatica
Monday 31st August, 2009

Abstract

Today organisations are facing growing pressures to be able to respond quickly to change. With the emergence of a global recession the need for agility has never been more present. As businesses continue to become more distributed and face-to-face contact becomes less frequent there has arisen a need for robust work flow tools that can handle the demands of today's dynamic businesses. It is also evident that nowadays businesses often rely on communication tools such as email to drive work flow, and there is much research which presents the drawbacks and limitations of heavy reliance on email. The aim of this research is to utilise the lessons learnt from the field of Software Configuration Management to support the Business Process Management field. The major objective for this thesis is to redesign current Issue Tracking Tools (predominantly used in Software Configuration Management) into a new workflow system known as *Traczilla*, which is capable of supporting more generic and dynamic business processes.

Contents

Part I

Introduction

Chapter 1

Problem Definition

1.1 Introduction

Software Configuration Management (SCM) has long been a significant area of research for developers. There are numerous challenges that software developers face in handling: feature requests, change requests, bug requests, versioning issues, and prioritisation of said issues. These challenges recently lead to the emergence of the 'Agile Manifesto' [13]. This Agile movement touted that "facilitating change is more effective than attempting to prevent it". From this, a new methodology for developing software was born. This agile methodology was used to great affect in promoting a configuration management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices that allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals. In addition, the emergence of this agile methodology resulted in the birth of a new set of Issue Tracking Tools that are designed to help Software companies keep track of and also prioritise their work[1]. These tools have proved to be most effective in streamlining the development of software and raising the overall quality of software.

Business Process Management (BPM) is another field which has garnered a significant growth in research in recent years. Organisations are realising the value in optimising their business processes. BPM promotes innovation, flexibility and integration with technology. As shown in Figure 1.1 below, one of the key building blocks of BPM is the Workflow Tool - also known as the Workflow Management System (WFMS). A WfMS is integral in stream-lining the business operations. A well designed WfMS is capable of driving the entire process from design to enactment and thereafter management or analysis. However, one of the drawbacks of using a Traditional WfMS is that they are generally limited in their ability to support "dynamic business processes". Dynamic business processes are best characterised by their un-structured, ad hoc nature, and often require human collaboration.

[1]E.g. Most notably, Trac (http://trac.edgewall.org/), Jira (http://www.atlassian.com/software/jira/) and Bugzilla (http://www.bugzilla.org/)

This problem has lead organisations to utilise a variety of workflow tools to suit their needs. Hence, despite widespread use of WfMS's, there are a wide range of businesses that have preferred to stick with more traditional workflow tools such as email. The motivation behind this is that many business processes are executed on an as needed or ad-hoc basis. Thus, flexible workflow tools such as email are ideally suited for this purpose. However there is much research which presents the drawbacks and limitations of heavy reliance on email [41, 19, 11].

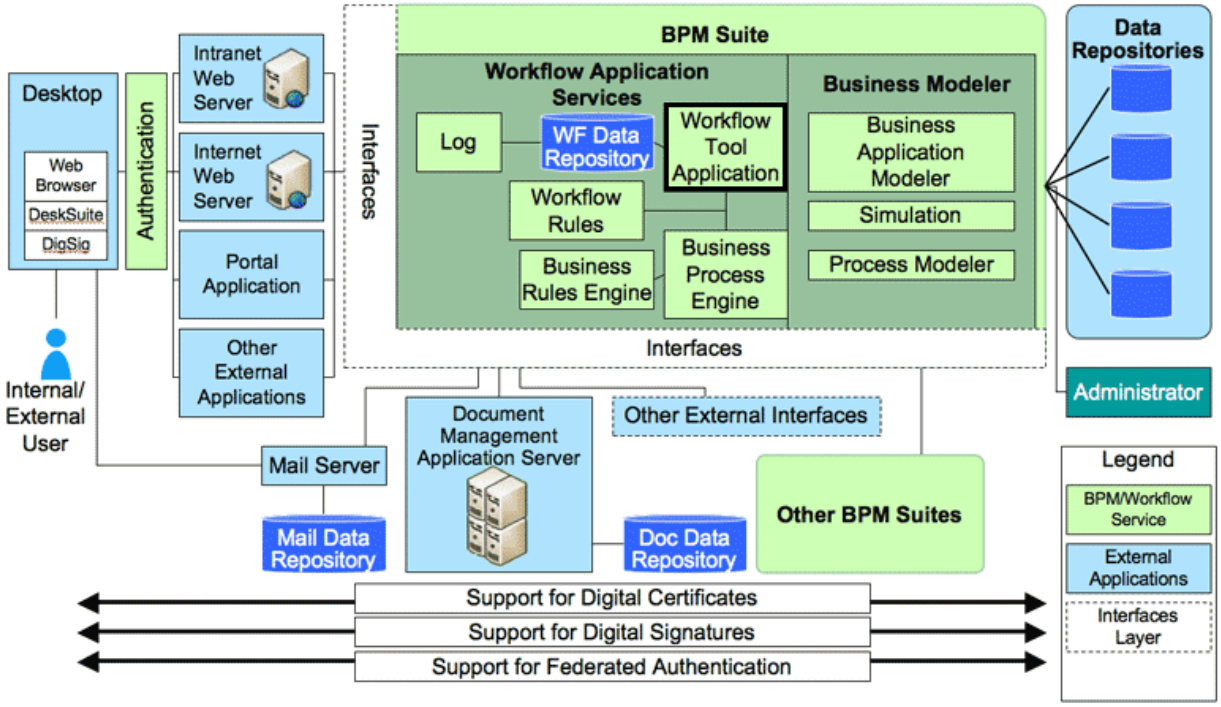

Figure 1.1: Business Process Management Suite, NIH (2007).

For the purposes of this paper, the definition and purpose of Traczilla is to build a workflow tool, that is an application that manages the process of passing information, documents and tasks from one employee (or system) to another. Looking at SCM we can see that Issue Tracking tools are in essence 'lightweight' workflow tools that can integrate, distribute and prioritise work (e.g. software bugs and issues). Helpdesks and Call Centers have been able to utilise similar Ticketing systems in order to handle the process of creating, updating, and resolving reported customer issues[2]. This adaptation immediately shows the broader applicability of theses systems. The major drawback that has prevented total adoption of these systems is the radical change it represents to customary work procedures. As many organisations grow accustomed to their way of doing business, it becomes difficult to change. For example, email is a workflow tool used by many organisations,

[2]E.g. OmniNet's OmniTracker Software (http://www.omnitracker.nl/) or Target Software's Internal Helpdesk (http://www.targethelpdesk.com)

and the incentives in drastically changing the way these businesses operate are simply not enough. An additional feature that would make these Issue Tracking tools more suitable would be to add Collaboration components to allow knowledge workers to interact and communicate whilst executing these dynamic business processes. Thus, businesses with dynamic processes could reap the benefits of traditional WfMS's by collecting structured data on each issue or ticket raised. The main challenge involves creating a tool that is not too heavy to slow down the process, whilst still being robust and lightweight enough to trace information on the entire process.

1.2 Research Relevance

1.2.1 Relevance to business

Today organisations are facing growing pressures to be able to respond quickly to change. With the emergence of a global recession the need for agility has never been more present. Many businesses are seeing rapid shifts in supply and demand, and this places further pressure on adjusting business processes. This demand for agility has been a necessity for quite some time. Organisations that find themselves unable to respond to change are left behind by the competition. The result is a need for robust Business Process Management Tools that can handle the demands of today's dynamic businesses. The emerging need for more and more businesses to be agile is not dissimilar from the events that sparked the Agile Manifesto. This represents an opportunity to explore the possibility of adapting Issue Tracking tools used in SCM to support more generic and dynamic business processes.

1.2.2 Relevance to science

The attempt to support dynamic business processes with SCM tools represents a venture into the unknown, and presents a difficult prospect that has not been effectively solved by modern-day Workflow Management Systems. While these systems have been a heavy research area for many businesses in recent years [37, 14, 3], there is still a prevailing problem, that is their inability to deal with the high volume of exceptions that arise in workflows within many of todays dynamic businesses. For example, many businesses today allocate resources based on incoming work/projects, which often means personnel need to be reallocated and resources redistributed (creating exceptions for modeled workflow). Thus, organisations using these very dynamic business models often find it difficult to accurately model all workflow possibilities and create an effective WfMS. Hence, a significant scientific contribution resulting from this thesis will illustrate how SCM technology can be utilised in order to manage these dynamic business processes.

1.3 Research Objective

The aim of this thesis will be to build a workflow tool, that is capable of managing the process of passing information, documents and tasks from one employee (or system) to another. Such a solution would provide organisations with a means to manage dynamic business processes where workflows cannot be cost effectively modeled accurately. Of course it should be noted that the immediate adaptation of Issue Tracking tools to business does not represent a new frontier in the research. As stated above, attempts have been made in the past which seek to directly utilise Issue Tracking tools within businesses, but the application of such attempts has been limited to very static businesses such as help desks and call centers. Therefore, a major aim of this thesis will seek to broaden the market model for these Issue Tracking tools by investigating the area of built-in collaboration tools.

> **Research Objective:** Develop a novel workflow management tool capable of supporting dynamic business processes, by adapting tools and techniques from software configuration management.

In order to accomplish the research objective, the following research questions shall be answered.

RQ I. *Why is current BPM Technology inadequate for managing dynamic business processes?*

1. What characteristics embody dynamic business processes?
2. What problems exist when handling dynamic workflow in current BPM Tools?
3. What are the limitations regarding Email communication?
4. What makes SCM Tools suitable for managing dynamic business processes?

RQ II. *What is the design criteria for a system supporting dynamic business processes?*

1. What functionality is available within current SCM Tools?
2. What additional functionality is needed to manage dynamic business processes?
3. What are the other design decisions regarding UI design, Database design and Back-end architecture?

RQ III. *What are the functional and non-functional requirements for the Traczilla System?*

1. What are the typical usage scenarios of Traczilla?
2. How does Traczilla accomplish the task of managing dynamic business processes?

1.4 Research Methodology

1.4.1 Research

As stated the main aim in this thesis project is to design and build a novel Workflow Tool, which is robust to the exceptions generated by modern day dynamic business processes. In order to build an effective and unique solution, the initial research will involve synthesising the reputable research in the areas of SCM and BPM, thereby exposing any similarities or differences between the two fields. After evaluating the current landscape of literature in the BPM field, a synthesis of the main issues, concerns and open research questions will be used to narrow the design focus, and to establish those key SCM components that can be utilised within the BPM field.

1.4.2 Design

Secondly, exploration and testing will be performed on a wide range of Off the Shelf (OTS) Issue Tracking Tools. An examination of the various strengths and weaknesses will be used to gauge which tool is the most suitable or the most adaptable for business purposes. Ultimately one tool will be chosen, adapted and extended for the purposes of this Thesis. From here, iterative modeling activities will be used to design-build-test each SCM component needed for the final system. Part of the design shall involve designing a scalable system. Therefore allowing smaller businesses to adopt Traczilla in place of current email driven work flow, and also allowing larger organisations to implement Traczilla to handle anomalies/exceptions not covered by their traditional WfMS. All code will be made available under an open source BSD license, to allow the community of open source software developers to extend and improve *Traczilla*. All modeling will be drawn in UML notation.

1.4.3 Develop

Finally, a hybrid development approach will be adopted for the purposes of developing Traczilla. A core focus will be on Agile principles to allow better adaptation to change. This agile approach will involve working iteratively through each phase of development and research. Testing will be done side-by-side development, this process will incorporate the idea of a continuous process, such that: code is developed in small releases, continuously integrated, and re-factored when necessary. However traditional procedures will also be adopted in that there will be a mandate for the production of a requirements document before code is written. This is to ensure that the design and scope of the system is set, before writing any code. This process will also incorporate a Review stage into each phase of development, such that bugs in the software can be removed earlier in the development process and thereby minimise maintenance work and eliminate big-bang testing processes from the end of development.

1.5 Research Scope

The description above gives a broad overview of the thesis. Obviously there is quite a bit of room for scope creep. This section will explain what the main focus of this research will be, and what will be considered outside of the research scope. The main research area will surround BPM with a focus on WfMS's. Background information on SCM and Agile methods will also be explored and explained. Additionally, research regarding CSCW (Computer Supported Collaborative Workware) and Email will be presented, as these elements offer key relevance for businesses.

On the development side, the workflow tool to be developed is intended to be a complete WFM solution. Thus, Traczilla will be a robust, light weight and stand alone support tool for businesses - allowing execution of business processes without getting in the way. The Collaboration tools used will be quite rudimentary most likely. The main idea is to explore the possibilities of what kind of communication is need by such a tool. As such it may be that not all Communication components will go into development.

After development is complete, an analysis of Traczilla will be performed in order to evaluate how Traczilla compares to previous solutions. However, due to the limited time available, and the lack of reliable real world data to test on it is unlikely that this analysis phase will be completely robust. A complete set of testing techniques will be performed in order to verify the correctness of the program. However, validating its suitability for business purposes will be left for a later study.

1.6 Research Outline

The thesis is structured in five parts and seven chapters, the first chapter and part being this Introduction and Problem Definition. In Part 2, we present Theoretical Background information in order to illustrate the wide-ranging literature and opinions in the fields of SCM and BPM. This part will be used to answer our first research question: "Why is current BPM Technology inadequate for managing dynamic business processes?". Chapter 2 will layout the conceptual framework for BPM. Here we will establish the characteristics that embody dynamic business processes. In addition, a survey of reputable, scholarly work on BPM tools will be given in order to establish the main issues, concerns and open research issues that exist in this field. Part of this chapter will present how many businesses utilise email as a complete WFM solution nowadays. In particular, a focus will be on establishing a number of the limitations of using Email, and how ineffective email is for the purpose of workflow management. In chapter 3, a synthesis of research on Software Configuration Management will be presented, in order to establish a conceptual framework on how business processes behave in Software development. An elaboration of research in the field of Agile Methods will also be presented for the purpose of detailing the factors that lead to this movement, which will be later compared with the factors pushing many of todays businesses towards the need for agility.

The Design of Traczilla will be illustrated in part 3, where we will answer: "What is the

design criteria for a system supporting dynamic business processes?". The design firstly aims to examine current SCM software and explore the functionality currently available. A summary of the main advantages and disadvantages we perceive will be shown in order to understand how these tools can be best adapted in order to support dynamic business processes. From here we will present the requirements that have been established for Traczilla, which will follow partly from the testing phase, and the conclusions from the literature review. Finally, a complete schematic design will be shown to visualise the database schema, UML modeling will be used to characterise the different methods and situations for which Traczilla might be used. For completeness, the back-end architecture will also be shown.

In part 4, we perform the analysis of Traczilla. Here we will answer the research question: "How does Traczilla accomplish the task of managing dynamic business processes?". In this part we will test Traczilla in order to firstly explore the underlying mechanisms of Traczilla, and to extrapolate any advantages and disadvantages compared to current workflow tools used in all fields. This will include system testing, and unit testing each newly developed component. Then we will closely examine one particular component in order to illustrate the key benefits we perceive for Traczilla. In part 5, the final part, we will summarise our findings and results. The implications of the research for both business and science will be given. Lastly, further research avenues will be highlighted.

Part II

Theoretical Background

Chapter 2

Business Process Management

This chapter illustrates the broad scope of Business Process Management and summarises the current landscape of applications available in the BPM front. We firstly define typical BPM terminology, and then highlight the essential goals of BPM. Thereafter, we narrow the scope and discuss a number of key workflow concepts. Following which, we present a series of workflow classifications. Finally, the current BPM technology will be introduced, along with the associated limitations in this technology.

2.1 Introduction

Business Process Management (BPM) is a "hot topic" within a number of modern-day research fields. BPM addresses the topic of process support from a broad perspective by incorporating different types of analysis (namely: simulation, verification, and process mining). Current research into BPM is divided between two communities: business administration, and computer science [39]. Business administration tends to focus on the ideas of increasing customer satisfaction, reducing costs, and increasing the efficiency of operations. Within the computer science community there are two factions. There are the researchers who have investigated formal methods for structuring and modeling real world business processes. Then there is the software community who have focused on providing robust and scalable software systems. We follow the latter research focus as we seek to implement a robust and scalable workflow tool based on SCM technology. However, within this chapter, we highlight the main contributions towards the entire BPM field.

2.1.1 Definitions

The basic concepts of workflow management can be best introduced using the definitions provided by the Reference Model of the Workflow Management Coalition, WfMC [20]. In spite of the efforts of the Workflow Management Coalition, the term workflow is still very fuzzy and used in many different contexts. Moreover, it is generally associated with the concept of business processes, which is also not very precise.

According to the WfMC, *workflow* is defined as "the automation of a business process, in whole or part" [20]. However, according to Davenport [9], a *workflow* is merely "a representation of the business process in a machine readable format". Where, a *business process* is "a set of one or more linked procedures or activities which collectively realise a business goal" [20].

A *workflow management system* (WfMS) is "a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications" [20].

Business Process Management (BPM) is a general term describing a set of services and tools that provide for explicit process: analysis, definition, execution, monitoring and administration. As such, BPM includes methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes. It can be considered as an extension of classical WfMSs and approaches [37].

Another important definition is that of a *knowledge worker*, which is a process participant that uses software systems to perform activities required to fulfil a business process. Equipped with detailed knowledge of the application domain, they perform the necessary activities which can not be automated [39].

2.1.2 Goals

BPM tools provide business users with the ability to model their business processes, implement and execute those models, and refine the models based on as-executed data. As a result, BPM tools are designed to provide transparency into business processes, and to centralise business process models in order to reveal execution metrics.

Thus, the most important goal of BPM is a better understanding of the operations a company performs and their relationships [21]. However, presently, business process flexibility is a strong driving force behind BPM. Flexibility is the ability to adapt to different circumstances. Where, "different circumstances" may be induced by changes in the market environment of the company [39].

2.1.3 Business Process Lifecycle

The BPM lifecycle concerns aligning processes with the organization's strategic goals, designing and implementing process architectures, establishing process measurement systems, and educating and organizing managers so that they can manage processes effectively [42].

The business process lifecycle starts with a definition of organizational and process goals, and an assessment of environmental factors and constraints that have an effect on the business processes of an organization. The iterative activities which constitute the business process lifecycle can be grouped into five categories: design, modeling, enactment, monitoring, and evaluation (refer to appendix A.1 for further details).

2.2 Workflow Classifications

In this section, we explore the different workflow classifications in the literature in order to determine the characteristics that define a "dynamic business process". The different workflow classifications will be discussed from a classical, and control-oriented view point. In order to demystify the confusion surrounding workflow terminology, we firstly present a widely accepted workflow classification which distinguishes between administrative, ad hoc, collaborative, and production workflows [28]. The basic parameters of this classification are task complexity and task structure as shown in Figure 2.1 below.

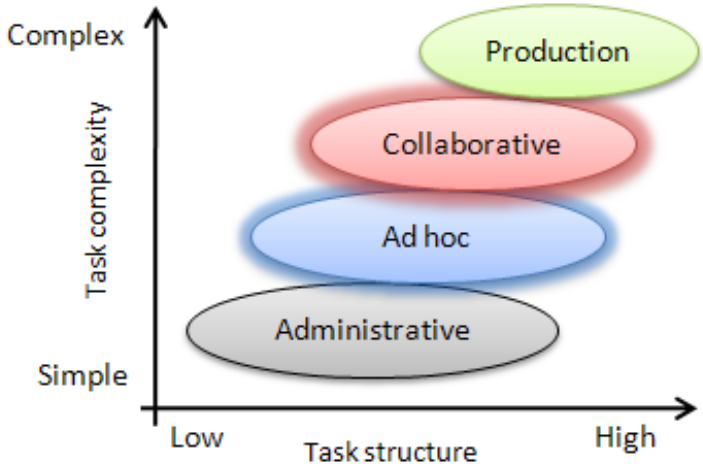

Figure 2.1: Classical Workflow Taxonomy, [3].

2.2.1 Classical Workflow Taxonomy

In general, *administrative* workflows refer to bureaucratic processes where the steps to follow are well established and there is a set of rules known by everyone involved [3]. Examples include: sending a report, authorising a request, or forwarding a work assignment - such workflows activities can be automated quite easily using document centered systems.

Ad hoc workflows are processes where the procedure is not completely defined in advance [14]. Ad hoc workflows typically require a rapid workflow solution, e.g., supporting the process of putting together the program of a professional conference. Ad Hoc workflows often have less repeatability, because some of the parameters needed to define the process are only known at run-time. This dynamic character causes many difficulties for traditional WfMSs in modeling and execution.

The third class of workflows, *collaborative*, is mainly characterized by the number of participants involved and the interactions between them. Unlike other types of workflows, which are based on the premise that there is always forward progress, a collaborative workflow may involve several iterations over the same step until some form of agreement

has been reached. Moreover, collaborative workflows tend to be very dynamic in the sense that they are defined as they progress. Since most of the coordination is done by humans, the workflow system is limited to the role of providing a good interface to communicate and record interactions, usually provided by e-mail [3].

Production workflows involve repetitive and predictable business processes, such as loan applications or insurance claims. Production workflows are highly structured processes with almost no variation. Traditional WfMS functionality is well suited to supporting production workflows [3].

2.2.2 Control-based Workflow Classification

Another characterization of workflow can be done based on collaboration and control - along a continuum from human-oriented to system-oriented [14]. On the one extreme, human-oriented workflow involves supporting human collaboration in order to perform the tasks of the business process. The requirements for WfMSs in this environment are to support the coordination and collaboration of humans. On the other extreme, system-oriented workflow involves automating software tasks with little or no human intervention. In addition to being highly automated, system-oriented workflows are very process-oriented.

2.2.3 Conclusion

In conclusion, from the literature above we find the dimensions along which workflow can be described, are as follows:

- *repetitiveness* and predictability of workflows and tasks;
- initiation and *control* of workflow, e.g., from human-controlled to automated;
- *collaboration* required between workflow participants;
- *structured* nature of the business process and associated activities.

This leads us to conclude that the characteristics that define a "dynamic business process" are: relatively in-frequent repetition, involving large amounts of human collaboration, and a generally unstructured nature.

2.3 BPM Technology

There is a lot of old, familiar technology that has contributed to the development of modern BPM systems. Many products in the IT arena support workflow functionality, including imaging systems, document management, electronic mail, groupware applications, legacy systems, and Business Process Reengineering (BPR) tools (see appendix A.2 for a more detailed overview).

In this section, we examine the current blend of workflow tools available to end-users and managers. These can be divided into four categories: Computer Supported Collaborative Workware (CSCW), Ad Hoc WfMS, Case Handling System, and Production WfMS. Each of these are shown in Figure 2.2 and discussed in detail below.

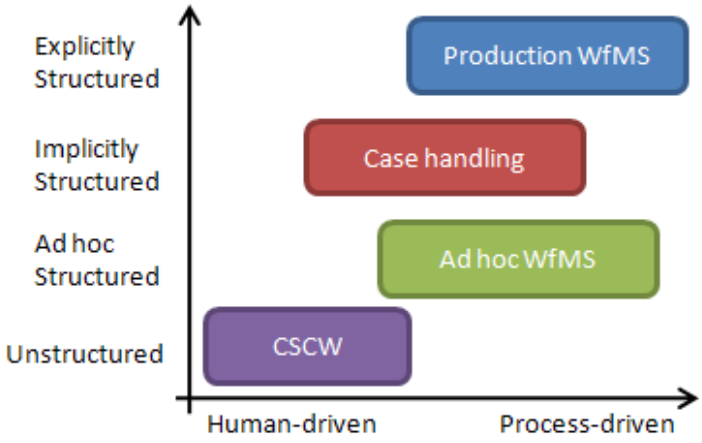

Figure 2.2: Four categories of Workflow Systems, [32].

2.3.1 CSCW

Computer Supported Collaborative Workware (CSCW), has received a significant amount of interest over the past decade. CSCW attempts to capture the inherent complexities of collaboration between people. It encompasses a broad listing of synchronous and asynchronous technologies including email, shared whiteboards, meeting schedulers, collaborative desktops, video conferencing, and other shared electronic media (as shown in Figure 2.3). In this type of system, users are allowed to perform their work in an 'ad hoc' way where solutions and goals are accomplished with whatever tools and data are immediately available [14]. The most successful has been the original technology of electronic mail.

As such, email has become an integral tool for many of todays organisations, and provides powerful facilities for distributing information between individuals within an organisation or between organisations. As a result, many businesses have now turned to "mail-centric" workflow technology, to handle ad hoc and collaborative workflow. This was postulated to be highly successful in its early days, because in these types of workflows, the WfMS is limited to providing a good interface to communicate and record interactions, making email an appropriate solution [3]. Especially when most of the coordination is done by humans . However, as we discuss in section 2.4.2 there are numerous undesirable side effects for utilising email in this unintended manner.

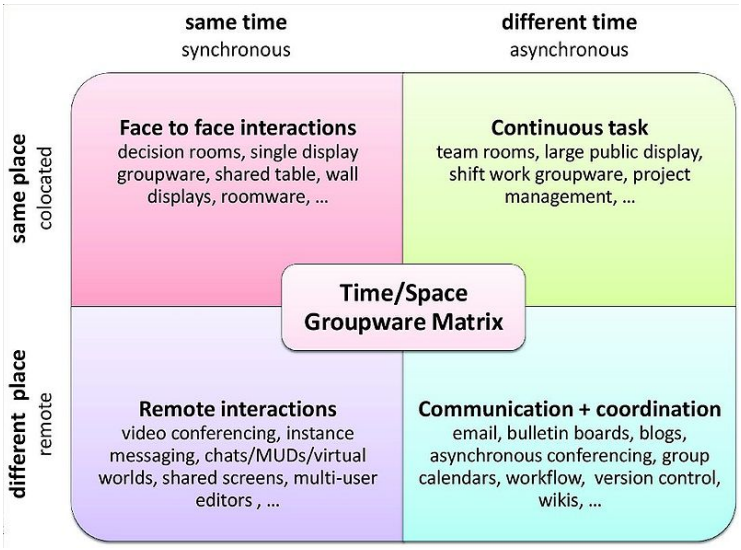

Figure 2.3: CSCW Matrix, adapted from [4].

2.3.2 Production WfMS

On the other end of the spectrum, production or traditional workflow management systems control the execution of business processes according to predefined (explicitly structured) workflow models. This approach is well suited to supporting static business processes that can be modelled once and thereafter executed in a routine fashion. However, in highly dynamic environments, there is a need to respond to new market requirements, not anticipated at build time, leading to new requirements regarding the flexibility of WfMSs.

It should still be noted that dynamic adaptations have been developed for traditional WfMSs. Exception handling was one such technique developed for the purpose of handling ad hoc deviations occurring at run time [37, 17]. Exceptions cover cases that deviate from the standard process definition. For example, an exception may occur when prerequisites for authorising a task are not met, or when there is incomplete or erroneous information in the task inputs/outputs [31]. However, since dynamic adaptability was not traditionally a topic in workflow management, the respective functionality currently available does not adequately support dynamic workflows [39].

2.3.3 Ad hoc Workflow Systems

Ad hoc WfMSs are new emergents in the workflow arena. They extend traditional WfMS functionality by allowing for dynamic changes of a single process instance. Such that, at any time the process definition for a case is structured, but the model can be changed while the case is being handled [32]. In Figure 2.2, this is referred to as ad hoc structured.

Ad hoc workflow systems allow for the on-the-fly creation and modification of process definitions. This functionality permits a workflow process to be instantiated without the complete process definition. Therefore, the traditional problems encountered when a pro-

cess definition changes can be avoided. Newer systems[1] also support workflow design by discovery such that, the routing of any completed process instance can be used to create a new template. This way, actual workflow executions can be used to create process definitions [29].

2.3.4 Case Handling Systems

On the Dutch workflow market, a new and interesting workflow system named case handling is emerging. The goal of a case handling system (CHS) is to overcome the limitations of traditional WfMSs. CHSs support implicitly structured processes. Moreover, they allow a case to move automatically through a subsequent state if all mandatory data elements are present [38].

The intention of a case handling system is to empower human actors in two ways. Firstly, it aims to give knowledge workers insight into completed and future activities. This limits the "context tunneling" effect, which is encountered by actors when a WfMS only provides a knowledge worker with the work items they are required to do. Secondly, a case handling system simplifies the handling of deviations from the regular process execution, which traditional WfMSs do not support adequately [35].

2.4 Evaluation of BPM Technology

In this section, we evaluate each of the above mentioned workflow technologies, and discuss some of their limitations - regarding their workflow classification applicability. Figure 2.4 illustrates the trade-off each system must make between *flexibility* and *support*, thus limiting the effectiveness of each individual system.

2.4.1 CSCW

CSCW systems offer a lot of flexibility but hardly any support for the processes. CSCW supports the execution of individual activities, but not the management and enactment of processes. Thus, when it comes to the support of work processes, we can rule out this technology. Pure CSCW systems are unaware of the processes taking place and, therefore, cannot be expected to offer effective process support. Hence, while these technologies are useful for overcoming communication problems over time and collaboration problems over distance, they lack the guidance and automation mechanisms needed to perform structured tasks [7].

[1] Such as InConcert (http://www.inconcertsw.com/prodinfo/welcome.htm), Ensemble and TeamWARE (http://www.teamware.us.com/) Flow

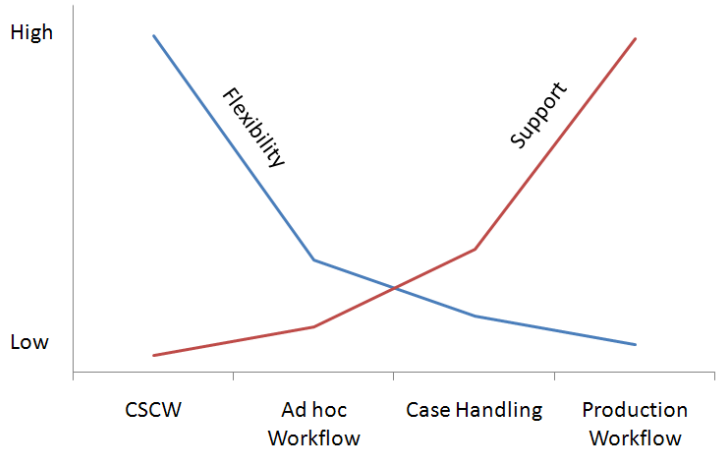

Figure 2.4: Business Process: Flexibility Vs Support, [32].

2.4.2 Email

As stated in the previous section, email is one of the most successful CSCW technologies, and indeed internet applications yet devised. However, empirical data shows that although email was originally designed as a communications application, it is now being used for additional functions, that it was not designed for, such as task management and personal archiving (information management). This usage has resulted in email overload [41].

Characteristics

Today, email has replaced the telephone as the main communication channel in the workplace [19]. Due to individual behaviors and workplace pressures, email has migrated from an asynchronous communication channel to a nearly synchronous communication channel. Many email messages are a manifestation of a user's participation in a business process. For instance, an employee in an organization with a centralized hiring process may receive automatically generated messages reminding them of an upcoming interview, requesting feedback on the candidate after the interview, and notifying them of the final decision. This has resulted in companies implementing email free holidays and employees filing for email bankruptcy[2]. These 21st century practices indicate a state of frustration with email so acute that users and organizations risk missing information [19].

While users may see this as a perfectly natural extension of the email client, the software was designed to function as an asynchronous communication channel, not as a "habitat" for personal information [11]. Using the email client for multiple functions has degraded the quality of the email communication channel and the user experience [41].

[2]Email free holidays (or bans) limit email traffic by asking employees to avoid email during a given time period, while email bankruptcy refers to individuals who are so overwhelmed by email that they can not read all of their email messages.

Email Overload

The main issue of using email in this fashion relates to email overload. When email overload occurs, email users have less time to spend on each individual email. Resulting in more emails to go through, more emails to action, and more information and tasks to manage. This vicious cycle increases the likelihood of missing an important email. In addition, email responses are typically less complete. Thus, the responses typically do not fully answer the correspondents' questions. As a result emails often have to be interpreted, increasing the possibility of miscommunication - as illustrated in Figure 2.5.

Figure 2.5: Circular effects of email overload, [19].

Email overload also creates problems for personal information management, because users often have cluttered inboxes containing hundreds of messages, including outstanding tasks, partially read documents and conversational threads. In addition, research shows that users experience major problems in generating appropriate folder labels when filing longer term information for later retrieval, and in reconstructing these labels when they engage in later retrieval. Furthermore, user attempts to rationalise their inboxes by filing are often unsuccessful. Consequently, important messages often get overlooked, or "lost" in archives [41].

2.4.3 Production WfMS

On the other end of the spectrum, production WfMSs offer a lot of process support but can only deal with situations that have been modeled explicitly. This creates numerous

challenges that remain for traditional WfMS, not least of which is the ability to deal with change. However, as this section points out, other limitations provide quite a hindrance also, including the lack of adequate support for knowledge workers they provide, as well as the limited process support for unstructured, human-oriented processes.

Dealing with Change

Adaptability has become one of the major research topics in the area of workflow management. As previously mentioned, today's workflow management systems have problems dealing with both momentary changes and evolutionary changes. As a result, traditional workflow management systems are not used to support dynamically changing workflow processes; or if they are, the workflow process is supported in a rigid manner, i.e. changes are not allowed or handled outside of the workflow management system [37].

Lack of adequate support for knowledge workers

Often the active selection of personnel by the WfMS has been deemed inadequate, for not taking into account the skills of the user, their competences, or their organisation positioning [39]. Structured process can in some ways streamline operations, but often process participants who have the experience and competence to decide on the required working procedures can perform the necessary business process activities in the most efficient way possible. For example, by skipping certain process activities the knowledge worker does not require or executing activities concurrently.

In addition, WfMSs provide very little room for creativity from the knowledge worker. Typical WfMS prescribe the process flow and ensure that the workflow is performed just as described, meaning any process instance that has not been envisioned by the process designer cannot be realised [39].

2.4.4 Ad hoc WfMS

Ad hoc WfMSs are attractive from a flexibility point of view. However, there are two important requirements for the successful application of ad hoc WfMSs. The first requirement is that actors are aware of the processes they are dealing with. This means that the processes should only be defined or modified by actors having a good overview of the whole process. The second requirement is that actors have the ability to use advanced modeling tools and have a good understanding of process definition techniques. It is essential that modelers can think in terms of sequential, parallel, conditional, and iterative routing. The two requirements often inhibit the application of this technology [37].

2.4.5 Case handling systems

Case handling systems (CHSs) aim at balancing process flexibility with data orientation (structure) to control the execution of business processes. Case handling takes into account

the active role of the knowledge worker by accepting their expertise and experience to drive and control the case. Whereas traditional workflow technology can be quite restrictive in this sense, by merely prescribing the activities required and their execution ordering. Thus, case handling offers much more flexibility to knowledge workers, while still providing well structured process support in order to maintain consistency and integrity [39].

2.4.6 Conclusion

Therefore, there is much overlap between the different workflow classifications and workflow technologies. As such, the following hierarchical classification is provided to illustrate this overlap:

- System workflows, typically implemented via a Traditional WfMS, incorporate:
  - Production Workflows: encompassing process-based systems; and
  - Administrative Workflows: encompassing document-centric systems.
- Human-oriented workflows, typically implemented via email, incorporate:
  - Ad hoc Workflows: encompassing ad hoc WfMSs and CHSs; and
  - Collaborative Workflows: encompassing CSCW technology.

The overlap between production workflows and system workflows, shows that these workflows are presently the easiest for traditional WfMSs to handle - because of their structure, automation, and repetitive nature. Secondly, the overlap between: ad hoc, collaborative, and human-oriented workflows, has lead many businesses to rely on email as a key workflow tool - because of their lack of structure, in-frequent repetition, and collaborative nature. Ad hoc WfMSs and Case handling systems lie some-where in the middle, with CHSs supporting more structure, automation, and repetition, while Ad hoc systems tend to support more in-frequent process instances with more collaborative and unstructured nature.

2.5 Summary

In summary, BPM is an iterative activity involving process: definition, analysis, execution, administration, and monitoring. However, an understanding of the nature of workflow can assist in understanding which of the business process lifecycle activities will add the most value. Presently, BPM tools are typically used for automating, measuring and optimizing business processes. However, as stated, the dynamic nature of todays workflows is pushing more organisations towards flexibility, transparency and traceability.

As a result, there have been a number of recent developments leading to new BPM technology that has overcome some of these problems. Case Handling systems in particular have overcome one of the most plaguing problems in the workflow management field - that

of an inability to handle complexly structured, non-repetitive - *human-oriented* workflows. Ad hoc WfMSs have also provided a step forward through the introduction of customisable workflows, allowing improved handling of *ad hoc* workflows. In addition, recent developments in CSCW has lead to a number of useful tool introductions that are now better able to handle *collaborative* workflows. However, a successful integration of such technologies has not yet been achieved to allow for complete support of dynamic business processes: i.e. ad hoc and collaborative (human-oriented, unstructured) processes. Thus, we examine the field of Software Configuration Management in the next chapter, in the search for a solution.

Chapter 3

Software Configuration Management

In this chapter, we present an introduction to the topic of software configuration management. As it is understood that readers may have limited knowledge about SCM and Software Engineering in general, the content to follow provides a broad overview of SCM, how it fits into the software development process, what its benefits are, and how it has evolved over time. We separate and highlight the traditional aspects of SCM, which are more aimed towards management, from the developer oriented aspects. The recent agile software movement in the software development community is also discussed, along with the repercussions this methodology had on SCM practices. After the background of these concepts has been established, further analysis about how these techniques can be utilised in BPM is illustrated in the final section.

3.1 Introduction

Software engineering is mainly concerned with the life cycle of software development. No matter what life cycle is chosen, be it: Waterfall, Iterative, Agile, etc., change is a constant feature of software development. To eliminate change is to remove the opportunities to take advantage of lessons learned, to incorporate advanced technology, and to better accommodate a changing environment.

Configuration Management (CM) is the discipline of controlling the evolution of complex systems [34]. CM first came into existence in the U.S. defence industry [25], where it was used to control manufacturing processes. Gradually, computers and software also evolved to the stage, where people were constrained to find ways to control their software development processes. Hence, SCM emerged in order to apply CM to the development of software systems [40]. Two ways in which SCM differs from general CM are: software is easier and faster to change than hardware, and second, SCM can (potentially) be more automated.

3.1.1 Definition

While there is no single definition of SCM, there are three widely disseminated views from three different sources: the Insititute of Electrical and Electronics Engineers (IEEE), The International Organization for Standardization (ISO), and the Software Engineering Institute (SEI). The most widely used description of the practices associated with configuration management is found in the IEEE Standard 828-1990, Software Configuration Management Plans [1]. It defines SCM as follows:

> "Software CM is a discipline for managing the evolution of computer program products, both during the initial stages of development and during all stages of maintenance."

The objective of SCM, is to ensure a systematic and traceable software development process in which all changes are precisely managed, so that a software system is always in a well-defined state at all times.

3.1.2 History

SCM emerged as a separate discipline in the late 1970s soon after the so called "software crisis[1]" was identified. SCM was adopted in Software Engineering (SE), in order to handle issues that were hampering SE development, like architecture, building, evolution, and version control [12]. The 1980's brought about the advent of tools such as SCCS (Source Code Control System), RCS (Revision Control System), Make (Automated Build Management), and Sablime (CM System). These tools targeted on specific functionality that is now known as either "version control" or "build management".

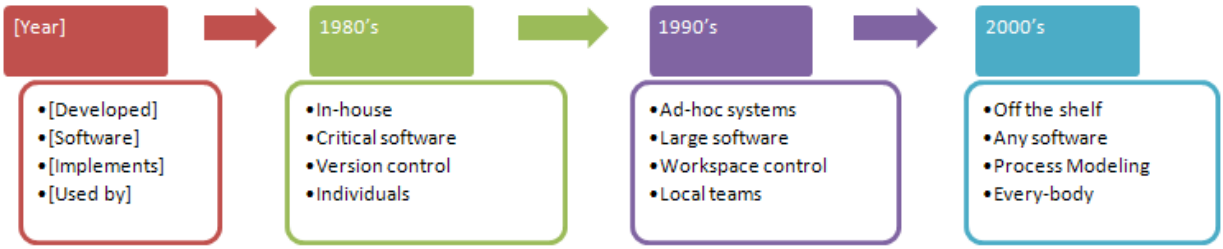

Figure 3.1: Evolution of SCM Systems.

As illustrated in Figure 3.1, the context and use in which SCM systems operate has changed significantly. First, SCM systems were originally used by a single person (the configuration manager) for managing critical software. The problem with early SCM systems is that they "helped the configuration manager, but annoyed everyone else" [12]. After considering the programmer as a major customer, a lot more useful tools started to emerge

[1]Period spanning the 1960s, 1970s, and 1980s, in which many software projects consistently ran over budget and schedule.

that actually began to help developers. This resulted in the emergence of versioning and build support, which was typically provided by some homegrown system. Use of SCM systems then changed to primarily supporting large-scale development and maintenance by groups of users. This resulted in a need for workspace management, which was quickly provided by newer, more ad-hoc SCM systems. Now, SCM systems manage the evolution of any kind of software by many different people in many, perhaps distributed locations utilizing many kinds of machines. This often requires explicit process support, which today's advanced, off-the-shelf SCM systems integrally provide. In fact, SCM is one of the few successful applications of automated process support [12].

3.1.3 Purpose and Benefits

SCM is a critical element of software engineering. SCM is needed because of the increased complexity of software systems, increased demand for software and the changing nature of software [25]. In addition, supressed visibility during the overall system evolution can result in a corresponding lack of management control [6]. It is also purported that SCM can be used as a strategic weapon that will give the organization an edge over those who are not using SCM or using it less effectively [25]. When used effectively during a product's whole life-cycle, SCM identifies software items to be developed, avoids chaos when changes to software occur, provides needed information about the state of development, and assists the audit of both the software and the SCM processes. Therefore, its purposes are to support software development and to achieve better software quality. As it can be seen in Figure 3.2, SCM is one of the major elements leading to better software quality.

Figure 3.2: Achieving Software Quality, [30].

Practicing configuration management in a software project has many benefits, including increased development productivity, better control over the project, better quality assurance, easier handling of software complexity, reduction in errors and bugs, faster problem identification and bug fixes, and improved customer goodwill.

3.2 Traditional SCM

In this section we discuss Traditional SCM activities. Traditional SCM is mostly about keeping control over the project; ensuring that it progresses according to schedule and that its delivery contains all the right parts. These activities are aimed more towards management. As stated above, the IEEE Standard 828-1990 is the authoritative standard on SCM, and it goes on to list specific activities associated with SCM:

> "SCM activities are traditionally grouped into four functions: (1) configuration identification, (2) configuration control, (3) status accounting, and (4) configuration audits and reviews."

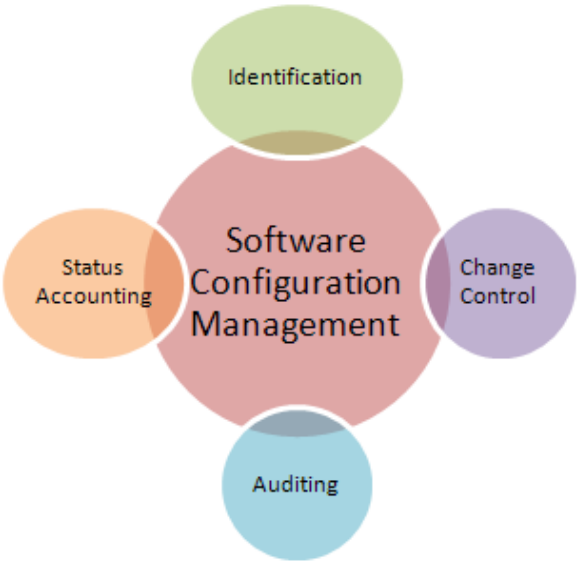

Figure 3.3: IEEE Traditional SCM Activities.

3.2.1 Configuration Identification

According to the IEEE standard, Identification involves: identifying, naming, and describing the documented physical and functional characteristics of the code, specifications, design, and data elements to be controlled for the project (Paragraph 2.3.1).

A more general and applicable description of Configuration Identification is the activity where a system is divided into uniquely identifiable components, called Configuration Items

(CI), for the purpose of making them unique and accessible [30]. Examples of CI may include: project plan, specifications, design documents, source code, test plans and test data, executables, make files, tools and the SCM Plan.

3.2.2 Configuration Control

According to the IEEE standard, Control involves: requesting, evaluating, approving or disapproving, and implementing changes (Paragraph 2.3.2).

Configuration Control is about handling changes in a controlled way through formal change control procedures, including: evaluation, coordination, approval or disapproval and implementation of changes to configuration items. A change request (CR) is a document containing a call for an adjustment of a system. Change can occur in the form of a change in: requirements, funding, schedule, customer expectations, or correction of an error/defect [18]. Given that software is something which is particularly easy to change, configuration control provides means to manage software changes in a structured, orderly and productive manner [24].

3.2.3 Status Accounting

According to the IEEE standard, Status Accounting involves: recording and reporting the status of project configuration items [initial approved version, status of requested changes, implementation status of approved changes] (Paragraph 2.3.3).

The aim of Configuration Status Accounting (CSA) is to keep managers, users, developers, and other project stakeholders informed about the various configuration stages and their evolution. This implies three basic tasks: data capture, data recording, and report generation [26]. Recording and reporting on the change process, requires traceability for all changes, this in turn requires the storage and maintenance of information about the:

- Product's configuration (such as part numbers or changes install in a given unit);
- Product's operational and maintenance documentation;
- SCM process (such as the status of change requests).

Status accounting reports include change logs, progress reports, CI status reports and transaction logs [25]. The information provided by the status accounting function is useful in determining the performance characteristics of the project, such as number of change requests, approval rate, number of problem reports, average time for a change resolution, average implementation time, and cost of implementing a change. This information will help when evaluating the performance of the project an when comparing difierent projects. Also, these details will help fine-tune the estimation and costing procedures of the organization, and may also help in identifying bottlenecks in the process or under-performing employees.

3.2.4 Auditing

According to the IEEE standard, Auditing involves: performing audits and reviews to determine to what extent the actual configuration item reflects the required physical and functional characteristics (Paragraph 2.3.4).

The objective of the Configuration Audit is to verify that the software system matches the configuration item description (requirements and standards) and that the system is complete [26]. Test reports and documentation are typically used to perform the audit. The process can be divided into three parts; Functional-, Physical-, and In-process Configuration Audit.

- A functional configuration audit aims to ensure that the software product has been built according to specified requirements. This process often involves testing of various kinds.
- A physical configuration audit determine whether all the items identified as a part of CI are present in product baseline.
- An in-process audit ensures that the defined SCM activities are bring properly applied and controlled.

3.3 Developer-oriented SCM

Developer-oriented SCM focuses on the developer and attempts to assist developers by automating the Traditional SCM tasks via specialised tools [25]. Automating manual SCM tasks provides more time to do the actual development work, leading to improved speed and productivity. A typical modern software configuration management tool provides primary services in the following areas [12]:

- Product Support:
  - Management of repository of components;
  - Release management.
- Engineer Support:
  - Workspace management;
  - Build management;
  - Version control.
- Process Support:
  - Change management;
  - Audit support;

– Generic process support.

A number of these tools are clearly irrelevant outside of the software development process. However, in the next section we will expand on those tools that have broader applicability. Firstly, showing how these tools support SCM activities.

3.3.1 Engineer Support

Workspace management

Given their critical role in the software development process, SCM systems must provide facilities through which other tools (and users) can interact with and manipulate the given artifacts. SCM systems implement workspaces to provide users with an insulated place in which they can perform their day-to-day tasks of editing and using external tools to manipulate a set of artifacts. Important considerations are whether workspaces must support distributed (or even disconnected) users, and how activities in independent ("parallel") workspaces are eventually integrated back in the SCM repository.

Version Control

The main purpose of version control is to manage different versions of configuration objects that are created during the software engineering process [30]. A Version Control System offers many advantages to both teams and individuals. It allows multiple developers to work on the same code base in a controlled manner, and it keeps a record of the changes made over time. The system also allows support for multiple releases (branches) of your software at the same time as you continue with the main line of development. All of which offers synonymous benefits associated with document control.

3.3.2 Process Support

Audit Support

As explained earlier, configuration auditing is the validation of the completeness of a product. SCM tools can automate most of the auditing, because they can generate the necessary information for verification purposes. For example, one person might need a history of all changes and another a log containing details about work completed. This data is organically collected by modern-day SCM tools.

Generic Process Support

Modern, high-end SCM systems push process support even further. They do not just support change control, but allow organizations to design and enforce general development processes throughout the enterprise. Exactly how this support is provided and integrated with SCM functionality is the subject of chapter 5.

3.4 Agile Methods

In this section we discuss, agile software development in general and how agile software development methods implement SCM. Many practitioners rejected traditional SCM systems because they were helping the configuration manager, and bothering everybody else. A common problem with traditional SCM is that there is is "too much process". Meaning the process can tend to slow down development, frustrate developers, limit customer options, and make for long integration times. A revolutionary step in the SCM field occurred with the emergence of the Agile Manifesto. The agile movement has gained significant popularity in recent years due to its "lightweight" low fuss approach [13].

Agile methods offer a number of benefits over traditional methods. By implementing a lightweight, customisable change request workflow, it allows traceability from business request to implementation [25]. In addition, agile methods are able to handle unstable requirements, and can deliver products in shorter time frames. Lastly, there is easier scope management, enhanced response to customers, better productivity, and increased quality [13]

3.4.1 Agile Manifesto

The term "agile" was coined in 2001, when seventeen process methodologists held a meeting to discuss future trends in software development. An agile process is said to be both light and sufficient. "Lightness" being a means of staying manoeuvrable. "Sufficiency" being a matter of supporting the people not the process. In consequence of this meeting, the "Agile Alliance" and its manifesto for agile software development emerged. The manifesto states that "experience has taught us we should value: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan" [13].

3.4.2 Agile SCM

Agile SCM acknowledges the reality of change, but suggests a change in methodology to deal with the uncertainty. As explained above, traditional SCM activities focus on control, and approach change in a prescriptive manner, setting detailed procedures and processes. Agile SCM takes the focus away from the process and focuses on boundaries and simple rules [13]. In essence, agile SCM is a well designed, light form of SCM that can be used as a less intrusive approach to SCM for software development projects. The approach requires direct involvement and interaction between users and customers, plus a development approach where functionality is delivered in small iterations. An iteration is a cyclical process that first involves the team negotiating new features or change requests with customers, and subsequently prioritise all work required. This process can however create what is known as a 'backlog', where any features or requests that have not been implemented are kept together and re-prioritised. In the next section we discuss the newly emerged Issue Tracking tools that promote Agile methods.

3.4.3 Issue Tracking Tools

Out of all the SCM tools introduced thus far, Issue tracking systems, which emerged as an Agile support tool, were one of the first to integrate the core features demanded by software developers. An issue tracking system (also called trouble ticket system, bug tracking system, or incident ticket system) is a computer software package that manages and maintains lists of issues, as needed by an organization. An issue tracking system often also contains a knowledge base containing information on each customer, resolutions to common problems, and other such data. These issue tracking tools, offer the potential to easily support ad hoc workflows. As they can support dynamic workflow which changes at run-time, and they allow on-the-fly creation of process definitions.

Issue tracking tools are currently used to great effect in software development. One of the major benefits to using such problem tracking tools is the analysis and reporting ability presented. Robert Grady's paper on Software Failure Analysis famously states that:

> "Software defect data is the most important available management information source for software process improvement."[15]

Grady of Hewlett Packard also points out that defect data can enable organizations to determine the weaknesses in their development processes and decide what changes they need to make and where. Another report on using web-based issue tracking tools for large software projects revealed that issue reports are extremely useful as they can be later analysed to reveal valuable info about why a decision was made on a particular issue [8]. In addition statistical reports can also be used to obtain useful information, e.g. regression analysis can often reveal the significant variables affecting the time to close off an issue [22]. This clearly represents useful business intelligence which could be used to make more informed decisions as well as improve work flow processes within any organisation. Using specialised software to manage problem reports, change requests etc has also been shown to be less error prone (as things can no longer *"slip through the cracks"* [16]) and the managing of such problems is also simplified by centralising all issue reports. A sample set of tools is further examined in Appendix B.

3.5 Bridging SCM to BPM

In this section, we illustrate how SCM concepts can be applied in a BPM context, and discuss some of the underlying business benefits of adapting SCM technology.

The "gap" as it stands in current BPM systems, surrounds supporting: ad-hoc, collaborative, human-oriented workflows in a manner that provides an underlying awareness of the business process. Whilst, still allowing improved flexibility, structure and transparency throughout the process. In the previous chapter, we examined the usage of email and found that many businesses now utilise email as a BPM tool. However, a number of limitations were also found in email, preventing sufficient support of the above mentioned workflows.

It is this problem that introduces a key area where SCM tools can be utilised to assist in the field of business process management, as is explained in the following paragraphs.

Firstly, the emergence of developer-oriented SCM tools brought about a surge of useful communication, collaboration and automation tools, capable of supporting dynamic and adaptive tasks. The introduction of these tools also moved the focus away from the manager and towards the developer. By moving the focus away from the manager, it removed the need for developers to complete redundant or irrelevant work items, and motivated a position of control. This in essence is analogous to many modern day WfMSs which focus on the manager as the key customer, by focusing on the "knowledge worker" (in this case) one can remove the burden of completing unnecessary tasks and can empower knowledge workers to take control of unstructured processes, whilst still providing automation, structure, and transparency to the manager.

Secondly, from the Agile Manifesto era, emerged a number of key support tools which now form the basis of many software development projects. Version control and workspace management first emerged as an engineers support tool for managing source code and development artifacts, these tools are also widely applicable in the areas of document/artifact management. Thereafter process support tools came into wide existence with capabilities for supporting process definition and mechanisms for verifying conformance, these tools are once again widely applicable in BPM. Change management tools followed, offering the ability to track changes with their associated impact on the system. Thereafter, Issue Tracking systems implemented an integration of these tools - thereby improve the handling of ad hoc and collaborative (human-oriented, unstructured) processes.

Lastly, the increased usage of SCM tools by larger teams who are in many cases in distinct locations has pushed the capabilities of SCM tools even further. The distributed usage of these tools governs another general concept that can be useful to a number of businesses, particularly as we see many organisations continue to globalise and distribute or outsource work. In addition, the primary objective of tracking business requirements from concept through implementation to customer delivery, is clearly shared among SCM and BPM.

3.6 Summary

In this final section, we summarise the main concepts behind SCM. CM emerged as a field for dealing with change and focused on control and discipline as two mechanisms for handling change. The evolution of SCM over the past three decades has moved the state of SCM away from strict control and discipline, and towards flexibility, traceability and tool automation. Traditional SCM activities include: configuration identification, change control, status accounting, and auditing. The onset of developer-oriented SCM brought about tools which offered the potential to automate these typical SCM activities. These tools came in the form of: Version control; Workspace management; Change management; Audit support; and Generic process support.

Each of these tools and principles aligns with the needs currently found in modern day

BPM Tools. That is, difficulty in handling: human-oriented, collaborative, and ad hoc/-dynamic workflows. Nowadays, SCM tools have expanded their usefulness and capabilities towards supporting responsiveness to customers, unstable requirements, generic process support and traceability from business request to implementation. In essence these goals are shared in common with many of todays businesses who demand agility and flexibility. In the Design chapters to follow we utilise a number of these tool features in order to build a BPM tool capable of supporting *dynamic business processes*.

Part III

Design

Chapter 4

Traczilla Design

In this chapter, we integrate the knowledge we have gained from previous chapters, in order to formulate a set of design criteria for Traczilla. We firstly gather together the commentary and criticisms on current BPM technology to establish the needs and trade-offs. We then explore the necessary user interface, database, and architectural aspects.

4.1 Design Decisions

To initiate the design considerations, we re-consider the current state of BPM solutions. After examining the limitations of current WfMS technology in chapter 2, we established four dimensions on which business processes can be distinguished: repetitiveness, collaboration, task automation and process structure. This allows us to classify a process as either system-centric or human centric. The current framework for systems supporting BPM is shown under this classification in Figure 4.1.

From the discussion in previous chapters, it is clear that current support for highly structured (production) processes (shown in the upper right-hand shaded corner) has evolved to a state where little advancement is possible. However, there is potential to solve problems in handing ad hoc and collaborative business processes (shown in the lower left). This has been accomplished in part through ad hoc WfMSs and Case handling systems. However, this support is still very limited and scalability can also be a problem as systems get exponentially more complex.

As such, the lower left corner has been predominantly supported by email - for which we have already established many limitations. The aim of Traczilla is to focus on the bottom left corner, i.e. design a new system intended for managing un-structured processes requiring large amounts of collaboration and human-centric behaviour. An approach which we purport is to integrate current technolgy used in CSCW and SCM. Therefore, a large part of the Traczilla design centers around combining various communication and workflow technologies into a single solution well suited to handling ad hoc, collaborative, human-oriented, unstructured processes.

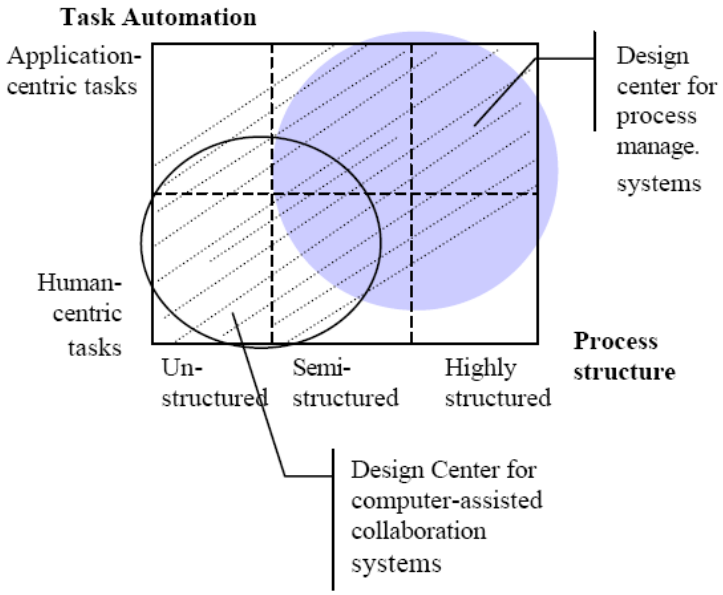

Figure 4.1: Current design centers in BPM, [10].

4.1.1 Workflow Support

According to Georgakopoulos [14], when designing a human-oriented workflow system, the main issues to address are:

- human-computer interaction;
- matching human skills to task requirements;
- changing office culture, i.e., how people need or prefer to work.

Human Interaction Workflows, require humans to be actively involved and to interact with information systems [39]. Flexibility is often required at run-time, because a person can be offered to work on a respective activity, and another person may in-fact select the activity and start working on it. Human interaction workflows require a work item list, where knowledge workers can then interact with the system using work item lists. These aspects will be directly considered in designing Traczilla.

4.1.2 Trade-offs

The design of any workflow system will require a number of trade-offs. Provided below is a list of the most significant trade-offs adapted from [33], along with the balance Traczilla seeks to achieve and the motivation for achieving this balance:

Support vs. Dictate

The workflow system must address the trade-off between supporting a user activity versus dictating the activity to them. The less leeway a worker has to improvise because of quality assurance, work standards, or artifact dependencies, the more the workflow system dictates the worker's activities. Given that, systems that dictate or automate, tend to clash with expert or knowledgable participants - Traczilla will act as a support tool for knowledge workers. Thereby, allowing knowledge workers to impart their knowledge on the process at hand and avoid "context tunneling".

Structure vs. Flexibility

Highly structured representations encourage automation, making it easier to maintain consistency. However, tolerating inconsistency increases flexibility in the evolution of the system because multiple competing or irreconcilable representations can be accommodated. Given that tightly constraining human activities may reduce the ability to accomplish tasks in a timely and appropriate manner - Traczilla will sacrifice some structure in order to provide added flexibility.

4.1.3 External Tool Integration

Tool integration is crucial for the design of Traczilla. Given that, a workflow system should specialize in delivering the right information to the right person at the right time. Once a person has the data artifacts at hand, the tools to produce, consume, manipulate, translate, annotate, or augment should be easily within reach as well. The goal of Traczilla is to seamlessly integrate the following knowledge worker tools.

Communication and Collaboration Tools

Execution of a workflow process involving many people inherently requires managing the interaction and communication between the individuals. Completion of ad hoc and collaborative activities often requires both formal and informal communication channels, which is beyond the scope of traditional workflow technologies. Thus, integrating CSCW tools such as: virtual whiteboards, wiki, workflow, and version control technology - should encourage participants to reach agreement on issues and to quickly communicate rationale and reasoning. The technologies of interest are categorised in Table 4.1 below.

Same Time/Different Place	**Different Time/Different Place**
Remote Interaction: - Video-conferencing - *Virtual whiteboards* - Messaging (Instant messaging, Email, and chat)	Communication + Coordination: - *Wiki* - Blogs - *Workflow* - Document/*Version control*

Table 4.1: CSCW Distributed Technologies.

Configuration Management Tools

Configuration management should be pervasive, but decoupled, in the Traczilla workflow system. As such, we examine three such tools from the SCM field: Trac, Bugzilla and Jira; in order to find which system can address current problems in handling dynamic and adaptable workflow most effectively.

The SCM Tools reviewed are among the most widely used in the software development community. All of which provide the following benefits: improved communication; increased product quality; improved customer satisfaction; status accountability; and increased productivity. Bugzilla, with an uninspiring user interface, is rich in features, but undeniably cumbersome to install and to maintain. Trac is a good, lightweight solution that combines excellent usability with plenty of flexibility. JIRA is a solid, powerful solution, providing almost all of the features of Bugzilla, and more, in an eminently more usable (and more productive) form - but at a cost. Table 4.2 below summarises the results of our examination (a complete review can be found in Appendix B):

	Bugzilla	**Trac**	**JIRA**
Synopsis	Open-source, powerful, flexibile, difficult to use	Open-source, lightweight, wiki-based, easy to use	Commercial, scalable, highly configurable
Search Functionality	Advanced query tool	Intuitive query builder	Full-text search
Change Management	Bug tracking, communication with team-members	Ticket system (bug-tracking, tasks, etc.)	Create and track any kind of issue in seconds
Workflow Support	Hard-coded	Minimal	Customisable workflow
Reporting	Simple bar/pie charts	Roadmap, Timeline for all recent activity	Real-time, relevant issue tracking reports
Miscellaneous	Excellent security, Optimised database for performance and scalability	Environment extensibility (via plugins), RSS feeds, iCal export	Designed with both business and technical users in mind

Table 4.2: Summary of features for each SCM Tool.

Upon review of the above tools, we selected *Trac* as a base for Traczilla development. Ultimately the decision as to which product to base the development of Traczilla on weighed up the following factors: availability (i.e. cost), usability (agile/flexible), and extinsibility. On the basis of this criteria *Trac* was the obvious choice, primarily because: 1) It is open source; 2) Motivates a collaborative approach rather than a more structured, traditional approach; and 3) Trac is easily extensible via plugins. In addition, Trac already integrates a number of the CSCW tools mentioned in the previous section (i.e. wiki, workflow, and version control technology).

4.2 User Interface Design

While the navigation interface provided by Trac is in general clean and minimalistic, some problems of redundancy and inconsistency can be identified. The goal of this section is to propose a clean-up to unify the navigation interface, and ultimately to establish guidelines for how Traczilla modules should hook into the navigation, so that they integrate nicely with the rest of the system. Trac currently provides the following navigational elements:

1. Project link: The project logo in the top left corner of every page;
2. Quick search: The search box in the top right corner of every page;
3. Meta navigation: The horizontal list of links just above the main navigation bar;
4. Main navigation: Primary means for switching between modules (Wiki, Timeline, Browser, etc);
5. Module navigation: Often rendered as horizontal list of links directly beneath the main navigation bar;
6. Local navigation: Navigational elements specific to the page currently being viewed. One example is the "Last change" in the Wiki module, another example are the "Previous/Next" links in the changeset module or ticket module;
7. Help links: Links to help documents (i.e. Wiki pages) relevant to the page currently being viewed;
8. Alternate formats: Links to alternate formats of the current page, such as the RSS feed for the timeline, or the comma-delimited text option for a report;
9. Footer links: Links to Edgewall and Trac.

Of this list, items 1, 2, 3, 4 and 9 are together referred to as the Global Navigation. The global navigation obviously should never change between different modules. The module- and/or page-specific navigation elements: 5 and 6, can together be referred to as the "Context Navigation". However, they are currently not clearly defined and their use is not consistent throughout the different modules.

For instance, in the Wiki, the module navigation (item 5 in the list above), contains the links Start Page, Index by Title, Index by Date and Last Change. These links are available independent of whether a page is being viewed or edited. While the first three of these links are "real" module navigation links, the fourth is actually page-specific, and is thus a local navigation link. Secondly, while browsing a directory or a file through the repository browser, the module navigation is empty, and the local navigation contains a link to the Revision Log and to the Last Change for the path being browsed. However, there should be a form that allows switching to a different revision by entering the revision number. Lastly, when viewing the list of available reports, two module navigation links

are available: Available Reports and Custom Query. Both remain available when opening a specific report. However, when viewing the report list or the custom query, only the alternative view remains selectable.

Possible Solutions:

Therefore, there is a need to fix the inconsistencies in the navigation for the release of Traczilla. Apart from layout details (such as the different look of local navigation links in the browser than in the reports module), there are three different options for resolving this problem:

- Draw a strong line between module-level and page-level navigation links. Develop a layout that helps the user understand the conceptual difference, so that he/she can learn to automatically look at the right place when searching for a specific link.
- Merge module-level and page-level navigation, as the conceptual difference may not be clear to the occassional user. Especially, given that more places to look for navigational elements generally means more confusion/frustration.
- Provide a user interface that will accept the knowledge worker as an important source to improve and control the process, by providing more user acceptance.

4.3 Database Design

4.3.1 Overview

The Traczilla database schema is largely built on the existing Trac database schema. The table below describes the tables utilised, rows highlighted in red represent additions or modifications required by Traczilla:

Table Name	Purpose
attachment	Descriptions of attachments (the files themselves are stored on disk).
auth_cookie	User login cookies.
component	Values that can be used in a ticket's "component" field, e.g. System (Accounting, HR, IT). The owner of a component is used as the default assignee for new tickets, if not overridden at the time of ticket submission.
enum	Maps integer IDs for issues' priorities, severities, etc. to human-readable names.
milestone	Used to specify Business processes/requirements along with the due dates.
milestone_struct	Milestone hierarchy to allow, business users to specify business processes at different levels.

node_change	1st half of the repository cache: for every changeset (as identified by the revision number), this table contains the nodes (i.e. files or directories) that have been added/modified/deleted.
permission	Text-based permissions specifying Username/action pairs. This table also stores the permission groups (e.g. Authenticated or anonymous).
report	Stores the SQL to generate canned database reports.
revision	2nd half of the repository cache: changesets, containing the revision number, author, time, and log message. The list of files and directories modified by a changeset can be found by joining with node_change on the rev column.
session	Last user visit time.
session_attribute	Information about user settings, including name, email, and diff options.
system	System information such as the database version and repository cache metadata
tkt_links	TicketDependency table used to link dependent tickets.
ticket	Ticket table storing all ticket related fields (e.g. Priority, owner, description, etc.).
ticket_change	Changes to tickets, on a field-by-field basis. The comment field associates a comment with a set of field changes.
ticket_custom	The values of custom ticket fields (e.g. Business Value or complexity).
wiki	Wiki pages (including old revisions).

Table 4.3: Traczilla database tables, adapted from [27].

Note: Please refer to Appendix C for the relational database schema.

4.4 Backend Architecture

Given that, Traczilla is based on the open source Issue Tracking Tool - Trac. In this section, we discuss Trac's architectural configuration. Trac uses a model-view-controller approach [2]. The controller is a Python class called a "module" which inherits from the Component class implementing the IRequestHandler interface. The controller reacts on user requests and prepares the data that will be used by a template engine to fill the adequate template, in order to render the view which will be sent back to the user.

4.4.1 Component Architecture

At the heart of Trac is a minimal component kernel that allows components to easily extend each others' functionality. At the same time, this component kernel also provides

a “meta-plugin-API” such that every component can easily offer its own plugin API by declaring “extension points” - as shown in Figure 4.2.

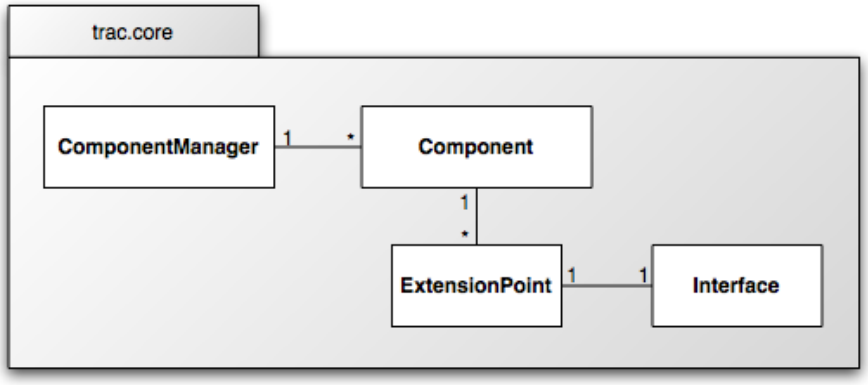

Figure 4.2: UML Trac Component Architecture, [2].

Public classes

The public classes for trac.core are defined below:

trac.core.ComponentManager Manages component life cycle, instantiating registered components on demand.

trac.core.Component Abstract base class for components.

trac.core.ExtensionPoint Declares an extension point on a component that other components can plug in to.

trac.core.Interface Every extension point specifies the contract that extenders must conform to via an Interface subclass.

4.4.2 Components

A Component is the basic building block in the Trac architecture. It is an object that provides a certain type of service within the context of the application. There can be at most one instance of any component. This implies that a component does not map to an entity of the application’s object model; instead, components represent functional subsystems.

Components can declare “extension points” that other components can “plug in” to. This allows one component to enhance the functionality of the component it extends, without the extended component even knowing that the extending component exists. All that is needed is that the original component exposes (and uses) one or more extension points - as shown in Figure 4.3.

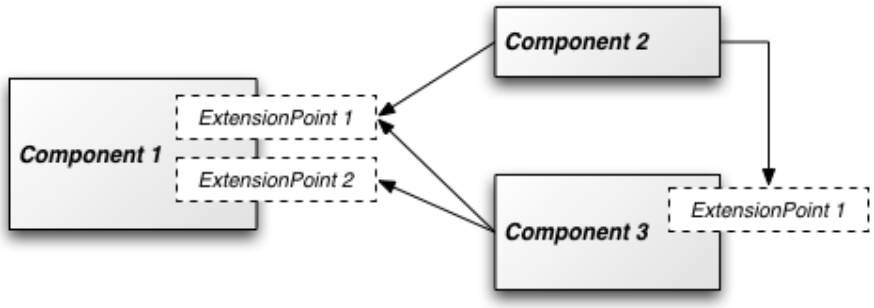

Figure 4.3: UML Trac Extension Points, [2].

A component can extend any number of other components and still offer its own extension points. This allows a plugin to offer its own plugin API (i.e. extension point). This feature is the basis of the plugin-based architecture employed by Trac. Meaning the actual functionality and APIs are defined by individual components. The component kernel provides the "magic glue" to hook the different subsystems together - without them necessarily knowing about each other.

4.5 Summary

The primary purpose of Traczilla is to inform users of the current problems, issues, and data needed to accomplish their particular tasks. Given that Traczilla is a tool designed for distributed organisations, allowing participants the ability to address outstanding issues with communication, collaboration, and remote interactions tools should aid in averting and resolving potentially work stopping issues. Therefore, integration of these tools into a workflow environment should aid continuous workflow progression and prevent bottlenecks to workflow. Hence, Traczilla will look at integration of messaging, wiki, version control and workflow technology.

In order to support dynamic and adaptive workflow Traczilla will allow tasks to progress at their natural pace while maintaining a model of the work being performed. It will also keep track of the status of workflows in order to maintain data and control transparency as well as keep assignments immediately viewable within the system via a Virtual Whiteboard. As with other policies, Traczilla will be decoupled from the implicit configuration management environment to allow for improved flexibility and customisability. Therefore, in order to provide the most robust and scalable solution, we perceive that the best approach is to combine existing technologies whilst striving to eliminate the drawbacks/limitations of each individual technology.

Chapter 5

Requirements Specification

5.1 Introduction

This is the requirements document for the Traczilla system. The requirements will be revised in the beginning of each iteration and when there is a need for it. The development of requirements can be seen at: http://smclaren.dyndns.org:8080/traczilla/. There is located the code base and test environment for the periodically updated requirements. The code base for this project is to be released under the open source BSD license.

5.1.1 Business Goals

The purpose of this project is to create a workflow system, which is pervasive enough to guide knowledge workers through predefined business processes, but lightweight enough to provide workspaces which merely support a knowledge workers activities. The system will operate as a web application so that users may access the system from distributed locations. Initially, the system is intended as a tool for small to medium sized businesses, but there are also applications in larger organizations. The goal is to create a workflow tool that is lightweight, dynamic, flexible and good looking like Trac itself.

The major benefits that this system provides are:

- Creates an environment in which knowledge workers can easily assign and delegate work items, and thereby formulate work models on the basis of successful execution;
- Capable of guiding knowledge workers through activities based on historical data;
- Allows creation of on-the-fly business processes (user stories), that can either be enforced to ensure structure and consistency, or relaxed to allow flexibility;
- Allows creation of events/tickets that are displayed on shared screens (Whiteboard), which can then be dynamically re-alocated to other users;
- Provides full workspace support to isolate each users work items.

5.1.2 Scope of Project

The intended project outcome is an integrated CSCW tool & workflow system, the main functionalities of which are shown in Figure 5.1, and listed below:

- Workspace management;
- Process support;
- Status accounting;
- Audit support;
- Search functionality;
- Administration and customisation ability;
- Communication and collaboration tools.

5.1.3 Main Domain Concepts

The main domain concepts from BPM and SCM that must be understood to gain a true context for the system, are shown below:

Concept	**Description**
Admin system	Web-interface to control the system.
Artifacts	One of many kinds of tangible byproducts produced during the execution of a business process. E.g., hand-off documents, budget projections, requirements, and design documents.
Authentication	Authentication is some kind of user authentication mechanism (e.g. IP, username & password).
Back-end	The part of the system that is not visible to the user (see front-end).
Backlog	Features, tasks or requests that have not been implemented or assigned (see triage).
Front-end	The part of the system that is visible to the user (see back-end). Also referred to as UI.
Issue	A ticket that is assigned to a person who must resolve it or reassign it to someone else.
RSS	Really Simple Syndication, an XML format which allows users to subscribe and easily monitor changes
Ticket	A file contained within an issue tracking system which contains information about a change, defect, issue, or improvement required.
Timeline	Historical audit trail of all changes made to the system.
Triage	Activity involving analysing a backlog of tickets in order to: group, discard, annotate, assign tickets.

User	User is the one who uses the system in some way. The user identity has been verified by authentication system.
User group	Different users can be grouped together based on common criteria (e.g. finance users).
User Story	Ad hoc business process formulated as a ticket in everyday business language.
Wiki	A database of pages that can be collaboratively edited using a web browser.
Workspace	Workspace is an isolated room, separating users different functionalities and work items.

Table 5.1: Domain concept definitions.

5.1.4 User Groups

Table 5.2 below, describes the intended users of the system:

User group	**Importance of group**	**Description**
Sysadmin	Medium	Highest level of administrator, can edit screen, edit/create users, and configure system settings.
Buinsess admin	High	Manages ticket system, including: workflow definitions and routing, ticket types, milestone definitions etc.
Knowledge worker	Very High	Uses the system to guide and direct work activities as well as to find information (a.k.a employee).
Manager	High	Uses the system to drill down and discover information.
Visitor	Low	Users who have not yet authenticated.

Table 5.2: Users of the system.

Note: there are two types of administrators: sysadmins who have full rights to everything and business admins who have full rights to customise the ticket & workflow system. By default, the visitor user group has no rights to perform or view any actions within the system. The "importance of group" refers to the frequency and criticality of each user groups intended actions.

5.2 System Overview

Building a workflow system for the purpose of structuring ad hoc and collaborative tasks requires a number of different technologies. Simply creating a single piece of software that

fulfills only the requirements of one workflow dimension would be ineffective in solving the underlying problems. By combining different existing technologies with new elements, this kind of system can truly help organizations add some structure and coherence to their unstructured processes. We recognize that users may not have all the required skills to utilize each particular technology effectively, therefore we concentrate on developing a system that is easy to use, customizable to suit the needs of any organization, and flexible enough to promote efficiency and guidance.

5.2.1 Workspace Management

The main purpose of workspace management is to provide the facilities through which other tools (and users) can interact with and manipulate given artifacts. This component shall provide users with an insulated place in which they can view and perform their assigned tasks, whilst allowing teams to assign tasks/tickets to fellow knowledge workers. Collectively, workspace management is made up of the following interacting sub-systems:

Ticket system

Provides simple but effective tracking of issues. As the central element of Trac, tickets are used for project tasks, feature requests, bug reports and software support issues. However, as the central element of Traczilla, tickets shall be used for collaborative tasks, user stories, and ad hoc issues. This subsystem shall be re-designed with the goal of making user contribution and collaboration as simple as possible. It should be as easy as possible to assign tickets, ask questions and suggest improvements.

Version Control

The *Trac Repository Browser* shall be used to browse directories and specific revisions of files stored in the repository of the configured version control system. This Trac component will be leveraged in Traczilla in order to provide built-in functionality for visualising "diffs" - changes to files.

5.2.2 Process Support

The main purpose of process support is to provide the tools and techniques to allow process definition and discovery:

Process definition

This shall create the process description in a computer processable form. The process definitions themselves should be flexible enough to enable partial process definition and execution. This requires information about user tasks to be undertaken, constituent activities and rules for navigating between them. These process definitions should be programmable

in multiple ways. That is, both visually and textually depending on the abstractions and level of expertise of the user.

Process discovery

Users shall be able to learn the structure of a business process from workflow log data. This learned structure should then be implementable as an efficient business process.

5.2.3 Status Accounting

The main purpose of the status accounting system is to keep managers, knowledge workers, administrators, and other stakeholders informed about the various stages of business processes and their evolution. This implies three basic tasks: data capture, data recording, and report generation.

Roadmap

Trac provides a view on the ticket system that helps planning and managing future software development milestones. Basically, the roadmap is just a list of future milestones. As part of Traczilla, the roadmap shall be used to provide a view on the ticket system that helps planning and managing of existing business processes (user stories). Hierarchical milestones will be added to enable organizations to sub-divide and aggregate tickets at different levels.

Statistics

Basic statistics based on usage and log data shall also be developed. This should illustrate: Ticket, Wiki, and SVN activity on a per user basis. Thereby indicating under-performing employees, and cumbersome or complex business requirements.

5.2.4 Audit Support

Trac supports logging of system messages using the standard logging module that comes with Python. The embedded audit support for Traczilla is designed to allow organizations to view the historical sequence of activities that have taken place. This requires traceability for all changes, which in turn requires the storage and maintenance of information from the: change logs, progress reports, transaction logs, and the underlying business process. To enable this support, Traczilla will utilize the:

Timeline

This Trac module provides a historic view of the project in a single report. It lists all events that have occurred in chronological order, a brief description of each event and if applicable, the person responsible for the change.

5.2.5 Search Functionality

Trac provides a simple but powerful full-text search functionality, which lets users search not only tickets, but also wiki pages and change-sets. The search also allows users to go directly to a change-set, ticket or report simply by entering its number. Another way to find information is through the creation of custom queries, which can be built using an intuitive query builder. This search functionality should be sufficient for the purposes of Traczilla.

5.2.6 Administration

The administration component of Trac is quite rudimentary. This functionality will be enhanced in Traczilla to provide supervisory functions which allow:

- *Customisation:* to enable business admin's to customize all components to suit organizations needs;
- *User admin:* to enable sysadmin's to create and manage fine-grained permissions for all users;
- *Workflow admin:* to enable managers to alter work allocation rules, to identify participants for specific organisational roles within a process, and to trace the history of a particular process instance.

5.2.7 Collaboration Tools

Traczilla shall utilise the following built in collaboration tools:

Wiki engine

Used for text and documentation throughout the system. This allows for formatted text and hyperlinks in and between all Traczilla modules. Editing wiki text is easy, using any web browser and a simple formatting system, rather than more complex markup languages like HTML. The reasoning behind its implementation is that HTML, with its large collection of nestable tags, is too complicated to allow fast-paced editing, and distracts from the actual content of the pages. Thus, the main goal of the wiki is to make editing text easier and encourage people to contribute and annotate knowledge for an organisation.

Virtual whiteboard

A 3rd Party plugin, which provides shared screen in which users can dynamically reassign tasks to other users, update the status of tickets, and "triage" backlogged tickets in order to add necessary data and assign to an appropriate individual. The whiteboard can also be used: to visualize and track progress; to group tickets according to their status; or to assign meta-information directly to items.

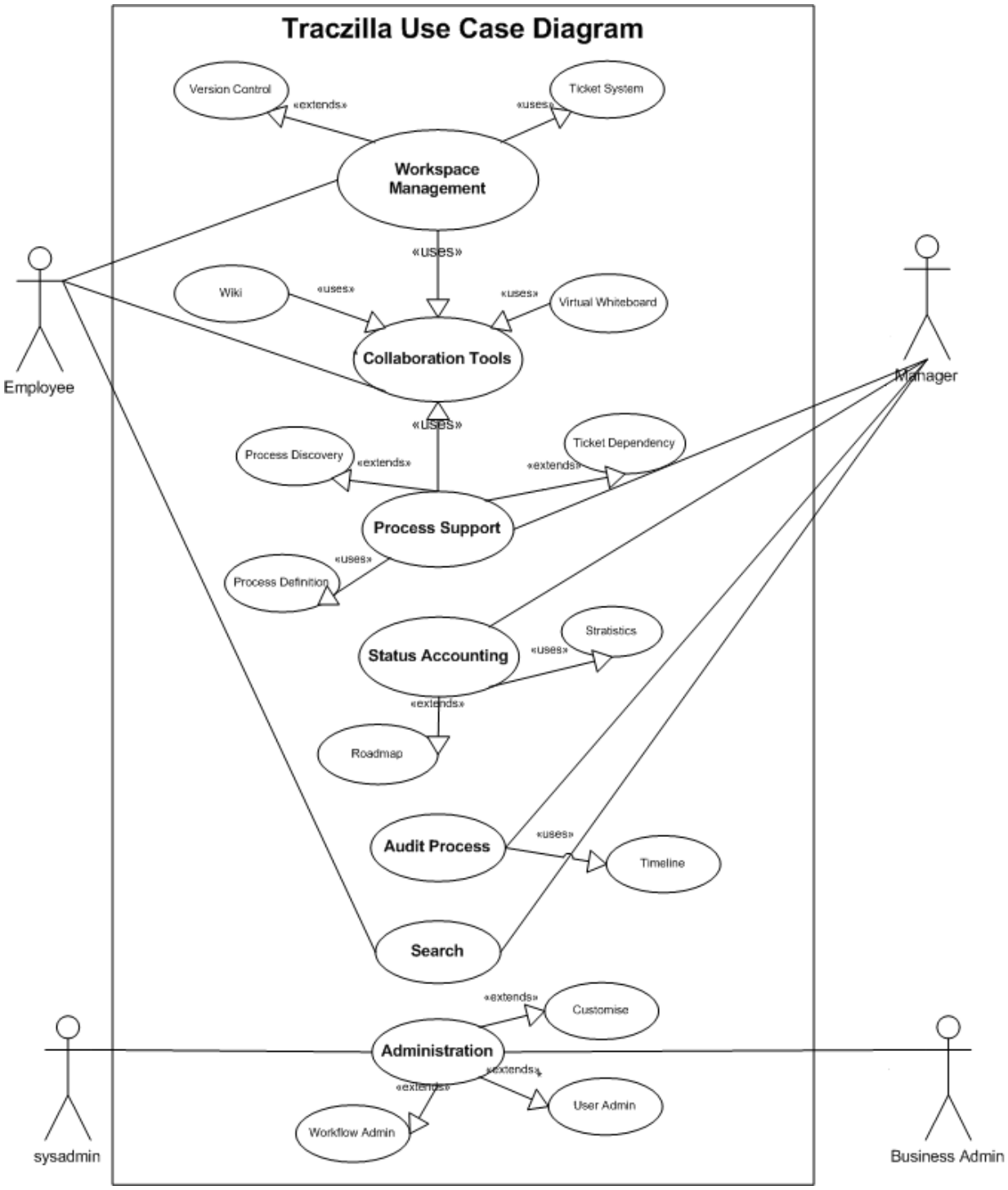

Figure 5.1: High-level Use Case Diagram.

5.3 Functional Requirements

This section describes the functionality of the system with use cases instead of a typical requirements list. The idea is that a working prototype will be produced in the first iteration and that prototype will be refined in later iterations.

5.3.1 Use Cases

We have used use cases to document the functional requirements. For additional details see section 5.3.2.

UC-001: Managing collaborative business processes

ID	UC-001	
Name	Manage collaborative business process	
Goal in Context	Allow knowledge workers to collaborate on tasks or business processes, by simultaneously manipulating artifacts, whilst allowing management to monitor and maintain control of progress through milestones.	
Actors	Knowledge worker (Employee)	
Pre conditions	Employees have appropriate access rights to Traczilla, and are able to access the Admin view in order to create milestones, submit tickets, and create artifacts.	
Basic Sequence	*Step*	*Action*
	1	Create milestone detailing Business Goal/Requirement (e.g. Increase profit margin by 20%) with due date.
	2	Create sub-milestone detailing the organisational process (e.g. Minimise overhead costs).
	3	Create child milestone detailing the operational process (e.g. Utilise JIT Inventory management).
	4	Create ticket [#1] with milestone set to "JIT Inv Mgmt".
	5	Employee accepts ticket and begins work.
	6	Artifact created (e.g. Purchase order), added to Subversion repository and linked to ticket [#1].
	7	Employee completes ticket, status changed to closed.
Post conditions	All milestones setup within Traczilla and linked to appropriate tickets, all artifacts maintained under version control and linked to appropriate ticket.	
Extensions	*Step*	*Branching Action*
	ALL	Utilise Whiteboard to review overall ticket and user story progress.
	ALL	Re-allocate tickets and update status of tickets as needed.
Sub-Variations	*Step*	*Branching Action*
	5'	Employee sets task status as pending/dependant on completion of another ticket [#2] (e.g. Sales order).

	5”	Search for dependant ticket, wiki item, or artifact.
	5”’	Drag and drop dependant items.
RELATED INFORMATION:		
Frequency of Use	Very High	
Priority	Must (Rank = 1)	
Effort	Very Large	

UC-002: Supporting ad hoc business processes

ID	UC-002	
Name	Support ad hoc business process	
Goal in Context	Minimise number of emails sent, and improve employee efficiency by allocating and prioritising ad hoc business processes (user stories).	
Actors	Employees, Managers	
Pre conditions	Employees have appropriate access rights to Traczilla, and are able to view and submit tickets to other employees.	
Basic Sequence	*Step*	*Action*
	1	Employee creates ticket of type User Story (with milestone defaulted to none).
	2	Ticket automatically entered into Story Backlog .
	3	Employee reviews and updates Backlog, necessary tickets added to user story - via Whiteboard.
	4	Employee annotates information onto story (e.g. business value and complexity), and assigns story to an operational business process (milestone).
	5	Traczilla prioritises work and employee changes status of each user story to reflect the work being done, and annotates appropriate information.
	6	Story follows its own lifecycle as per workflow configuration (see UC-004).
	7	Employee completes user story and adds final notes to the ticket.
	8	Status of ticket/user story changed to closed.
Post conditions	All stories set up through Traczilla, all tickets linked to user story, with tasks annotated and information disseminated as necessary.	
Extensions	*Step*	*Branching Action*
	5a	Employee rejects task
	6a	Management requests report on all user stories that are in progress at the moment (see UC-006)
	8a	Manager reviews complexity and business value points of User Story compared to child tasks.
	8b	Due lessons learned annotated as comment to the user story.
Sub-Variations	*Step*	*Branching Action*

	6'	Employee sets story status as pending on completion of another user story.
	7'	All user stories can be edited, annotated, re-assigned, prioritized and discussed at any time via Whiteboard.
RELATED INFORMATION:		
Frequency of Use	High	
Priority	High (Rank = 2)	
Effort	Large	

UC-003: Supporting execution of work items

ID	UC-003	
Name	Support execution of individual work items	
Goal in Context	Increase efficiency of administrative processes/work items by collectively assigning tickets and automatically prioritizing tasks.	
Actors	Employees, Managers	
Pre conditions	Employees have appropriate access rights to Traczilla, and are able to view and submit tickets to other employees.	
Basic Sequence	*Step*	*Action*
	1	Employee creates ticket of type Work Item (with milestone defaulted to none).
	2	Ticket automatically entered into Task Backlog.
	3	Employee reviews and updates backlog (tickets assigned to employees).
	4	Employee annotates information onto ticket (e.g. Business value and complexity), and assigns task to a user story.
	5	Traczilla prioritises work and employee changes status of ticket to reflect the work being done, and annotates appropriate information.
	6	Ticket follows its own lifecycle as per workflow configuration (see UC-004).
	7	Employee completes ticket and adds final notes to the task.
	8	Status of ticket/task changed to closed.
Post conditions	All tickets assigned through Traczilla, all work appropriately prioritised, with tasks annotated and information disseminated as necessary.	
Extensions	*Step*	*Branching Action*
	5a	Employee rejects task
	6a	Management requests report on all tickets that are in progress at the moment (see UC-006)
Sub-Variations	*Step*	*Branching Action*
	6'	Employee sets task status as pending on completion of another task.
	7'	All tickets can be edited, annotated, re-assigned, prioritized and discussed at any time via Whiteboard.

RELATED INFORMATION:	
Frequency of Use	High
Priority	Must (Rank = 3)
Effort	Minimal

UC-004: Process support

ID	UC-004	
Name	Process support	
Goal in Context	Provide tools and techniques to manage process definition and discovery.	
Actors	Business admin	
Pre conditions	Business admin has appropriate access rights to Traczilla's Admin view and can create ticket types, and workflow definitions.	
Basic Sequence	*Step*	*Action*
	1	Create new Ticket type to house workflow definition (e.g. ad hoc process).
	2	Define workflow for "ad hoc process" (states, transitions, permissions, default actions).
	3	Create new ticket of type "ad hoc process".
	4	Ticket follows lifecycle as defined by workflow definitions.
	5	Ticket completed, status changed to closed.
Post conditions	Ticket follows pre-defined workflow definition, however knowledge worker may deviate from process if required.	
Extensions	*Step*	*Branching Action*
	3'	Knowledge worker deviates from workflow definition in order to accomplish process more efficiently.
	3"	Appropriate notes annoted to ticket, with reason for deviation.
Sub Variation	*Step*	*Branching Action*
	1a	Create new user story.
	1b	Create tickets associated with user story.
	1c	Create dependency between tickets and user story.
	1d	Examine dependency graph to discover implicit business process.
RELATED INFORMATION:		
Frequency of Use	Medium	
Priority	Intermediate (Rank = 5)	
Effort	Large	

UC-005: Customise system

ID	UC-005	
Name	Customise or Administer system	
Goal in Context	Customisation of all possible components, allowing drop in and removal of fields.	
Actors	sysadmin	
Pre conditions	sysadmin has TRAC_ADMIN permissions to Traczilla, and are able to access the Admin view and customize all system settings.	
Basic Sequence	*Step*	*Action*
	1	Create custom ticket field (e.g. Client).
	2	Customise UI navigation (e.g. Logo, Navigation links, etc).
	3	Define more/less Business Process levels i.e. Milestones.
Post conditions	All customizations made through Traczilla's web interface.	
Extensions	*Step*	*Branching Action*
	1	User administration activities (add/remove users, assign/revoke permissions etc.).
	2	Change log level settings.
RELATED INFORMATION:		
Frequency of Use	Low	
Priority	Medium (Rank = 6)	
Effort	Intermediate	

UC-006: Generate management reporting data

ID	UC-006	
Name	Generate management reporting data	
Goal in Context	Keep managers, knowledge workers, administrators, and other stakeholders informed about the various stages of business processes and their evolution.	
Actors	Managers	
Pre conditions	Manager has REPORT_VIEW permissions to Traczilla, and are able to access the Statistics view.	
Basic Sequence	*Step*	*Action*
	1	View progress of each business process/milestone.
	2	Aggregate tickets targeted for each milestone, and calculate the ratio between active and resolved tickets - displayed as a milestone progress bar.
	3	Group progress by: Business Value, Ticket Completion, Complexity.
	4	Filter on Ticket type and Milestone type.
Post conditions	Managers are aware of any bottlenecks within various business processes, and where the most value add is being gained.	

Extensions	*Step*	*Branching Action*
	1	Create interactive charts showing: Ticket/Wiki/SVN activity per user.
	2	Manager creates custom statistical charts based on tickets per milestone/user/status.
RELATED INFORMATION:		
Frequency of Use	Medium	
Priority	Medium (Rank = 4)	
Effort	Intermediate	

5.3.2 Attributes for Entities

The following properties are information that the user can enter or interact with when using the system. Some of these attributes are in close relation with the overall architecture or with the contents. Also, some of the following information is optional for the basic system functionality but it is put here to ensure the system's extensibility.

Ticket attributes

- Related use cases: UC-001, UC-002, UC-003, UC-006
- Required information:
  - Reporter - The author of the ticket.
  - Assigned to/Owner - Principal person responsible for handling the issue.
  - Summary - A brief description summarizing the problem or issue.
  - Description - The body of the ticket, should be specific, descriptive and to the point.
  - Type - The nature of the ticket (for example, user story or work item)
  - Status - Current status of issue, one of: new, assigned, closed, reopened.
  - Priority - The importance of this issue, ranging from trivial to blocker.
- Optional information:
  - Business Value - Numeric estimation of the business value added by ticket.
  - Complexity - Numeric estimation of complexity in resolving ticket.
  - Milestone - When this issue should be resolved at the latest.
  - Component - The department or subsystem this ticket concerns (e.g. Accounting).
  - Keywords - Keywords that may be useful for searching and report generation.
  - Cc - A comma-separated list of other users to notify. (Note: this does not imply responsiblity.)
  - Resolution - Reason for why a ticket was closed. One of fixed, invalid, wontfix, duplicate, worksforme.
  - Blocking/Blocked By - Comma separated list of dependent tickets

Milestone attributes

- Related use cases: UC-001, UC-002
- Required information:
  - Name - Unique name to identify milestone.
  - Due - Deadline date to accomplish milestone.
- Optional information:
  - Parent - High-level milestone to which this milestone relates (blank if Business Goal).
  - Type - Implicit variable based on level of milestone (either: Business Goal, Organisational Process, or Operational Process).

Whiteboard attributes

- Related use cases: UC-001, UC-002, UC-004
- Required:
  - Name of the ticket, description, dependant milestone.
  - State (New/Assigned/Accepted/Closed): display as swimlane.
  - User story view: aggregate on business process level.
  - Work item view: aggregate on milestone.
- Optional:
  - Display as a movable sticky note field
  - Color coding or iconic flags shall be applied to items to distinguish them.

Permission attributes (for the authentication support)

- Related use cases: ALL
- Required information:
  - User: Username which permission applies
  - Permission: Text-based permission in the form MODULE_PRIVILEGE. For example:
    - TICKET_CREATE: Create and assign new tickets
    - WIKI_MODIFY: Change wiki pages
    - REPORT_VIEW: View reports and statistics
    - TRAC_ADMIN: Perform any operation
- Optional information:
  - Group: User group which permission applies (e.g. Authenticated or Anonymous)

Log attributes (for audit support)

- Related use cases: UC-001, UC-002, UC-003, UC-004, UC-006
- Logging methods: can be set via web interface (log_method)
  - none: Suppress all log messages.
  - file: Log messages to a file, specified with the log_file option in trac.ini.
  - stderr: Output all log entries to console (standalone server only).
  - syslog: (UNIX) Send all log messages to the local syslogd via named pipe /dev/log. By default, syslog will write them to the file /var/log/messages.
  - eventlog (Windows) Use the system's NT Event Log for Traczilla logging.
- Log levels: can be set via web-interface (log_level):
  - CRITICAL: Log only the most critical (typically fatal) errors.
  - ERROR: Log failures, bugs and errors.
  - WARN: Log warnings, non-interrupting events.
  - INFO: Diagnostic information, log information about all processing.
  - DEBUG: Trace messages, profiling, etc.
- Changeset messages: (artifact management)
  - Timestamp - When the changeset was commited
  - Author - Who commited the changeset
  - Message - A brief description from the author (the commit log message)
  - Files - A list of files affected by this changeset

Timeline attributes (for audit support)

- Related use cases: UC-001, UC-002 UC-003, UC-006, UC-007
- Required information:
  - Outline - brief excerpt of the actual event (comment or text, if available).
  - Hyperlink - to the specific event in question
  - Wiki page events - Creation and changes.
  - Ticket events - Creation and resolution/closing.
  - Artifact changes - Repository check-ins.
  - Milestone - Business process completed
  - Status - What is the current status? One of new, assigned, closed, reopened.
  - Priority - The importance of this issue, ranging from trivial to blocker.
- Optional information:
  - Ticket events - Other ticket changes, such as comments, status change, owner change.

5.4 Non-functional Requirements

Non-functional requirements are documented in this section, based on [7].

ID	Requirement	Importance	Effort	Related Use Cases
N1	*Adoptability* - system should integrate easily without burdening existing work cultures or technical infrastructures.	Medium	Low	UC-001, UC-005
N2	*Flexibility* - workflow system should allow for on-the-fly definition of business processes. Artifact routing should allow knowledge workers to drive process, or use the system for guidance. UI should provide different views for different stakeholders.	High	High	UC-002, UC-005
N3	*Extensibility* - system at large should be extensible to encourage and allow new components after deployment - allowing on-going applications in new domains. Workflow infrastructure should also be extensible, by providing the ability to execute empty process definitions or map out complete process definitions.	Must	Medium	UC-004 , UC-005
N4	*Search-ability* - system should allow users to quickly find information that is relevant to their task or activity.	High	High	UC-003, UC-006
N5	*Lightweight* - User interface components, workflow infrastructure, and artifact tools should be maneuverable to encourage customisation and provide incremental scaling from small to large organisations.	Must	High	ALL
N6	*Business Process Discover* - Dependencies between tasks/artifacts/wiki pages should be possible to allow users to discover implicit business processes or patterns.	Must	Low	UC-001, UC-002, UC-004

Part IV

Analysis

Chapter 6

Traczilla Analysis

6.1 Introduction

In the previous chapter, we discussed the high-level functionality and requirements of Traczilla at large. We described the pre-existing Trac components that would be directly utilised in Traczilla. In this chapter we provide a deeper analysis of each plugin developed specifically for the purpose of enhancing Traczilla's functionality. We focus our testing effort on these newly developed Traczilla plugins. We also provide a real world scenario in which Traczilla would be a suitable tool to manage a dynamic business process. We start with a description of each plugin, along with the use cases to which they relate. After which we run unit tests on each component/system implementing new features. In the last section, we provide a detailed analysis of the WorkflowEditor plugin.

6.2 Plugin Descriptions

In order to understand where and how each plugin is utilised within Traczilla - we first present a functional hierarchy showing the underlying arrangement of each component. In total, we developed 6 plugins for the Traczilla system: IniAdminPlugin, TicketDependencyPlugin, WorkflowEditor, CustomFieldPlugin, MultipleWorkflowPlugin, and the StatisticsPlugin. Figure 6.1 below shows how each of these plugins fits into the system at large.

6.2.1 IniAdminPlugin

This plugin allows a sysadmin to customise Traczilla via the Web Admin API. It uses the Administration panel available in Trac 0.11 to allow modification to any field exposed through the trac.ini configuration file. This currently includes all core Trac settings including: UI navigation, database settings, log levels, etc. It builds on the following trac extension points: trac.admin.api, trac.util, trac.web.chrome, and trac.config.

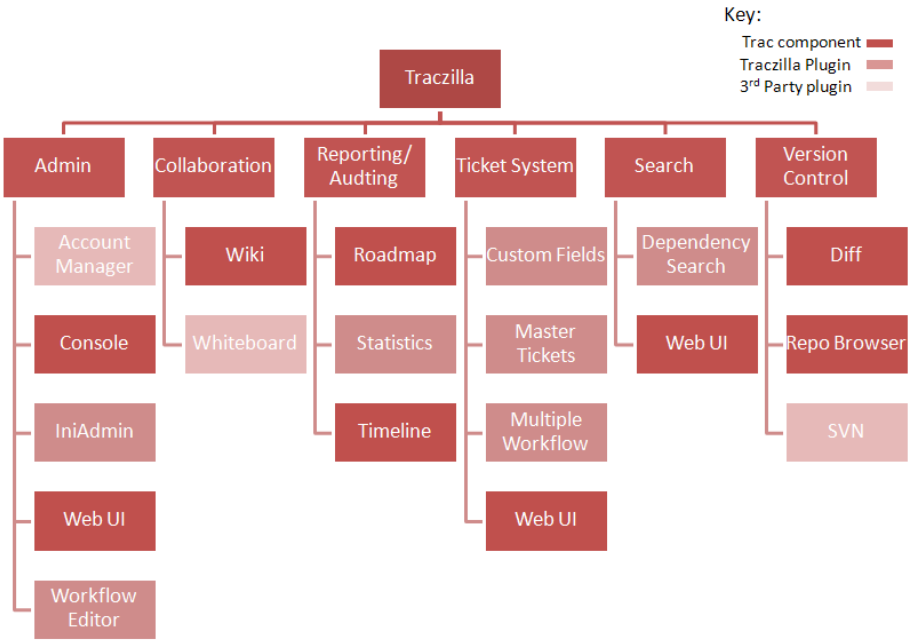

Figure 6.1: Traczilla functional hierarchy.

Implementing the: ITemplateProvider, and IAdminPanelProvider - through the methods: `get_admin_panels(self, req)` and `render_admin_panel(self, req, cat, page, path_info)`.

6.2.2 WorkflowEditorPlugin

The Trac issue database provides a hard-coded workflow that all tickets must follow. In order to overcome this limitation, the WorkflowEditorPlugin was developed to allow business users to customize the workflow of tickets to their organizations needs. This plugin provides a simple web interface that allows users to edit the ticket workflow. Thereby, providing a means to define on-the-fly business processes. This plugin also utilizes the 3rd party component: GraphViz to draw the workflow graph. It implements the IAdminPanelProvider, and IRequestHandler from the trac.admin and trac.web API's respectively. Further examination of this plugin is provided in section 6.4.

6.2.3 TicketDependencyPlugin

This plugin adds "Refers to" and "Referred by" fields to each ticket, enabling users to express dependencies between tickets, wiki pages, and artifacts stored in the Repository browser. It also implements a graphviz-based dependency-graph feature, allowing users to visually understand the inter-dependencies between tickets - thereby providing a step towards Business Process Discovery. The dependency graph is viewable by clicking 'dep-

graph' in the context (in the upper right corner) menu when viewing a ticket that refers to or is referred by another ticket.

This plugin utilizes the 3rd party component: GraphViz to draw the ticket dependency graph (as shown in Figure 6.2). And adds the table: `tkt_links` to the Traczilla database schema. It implements the IRequestFilter, and IRequestHandler from the trac.web API, through the following methods: `post_process_request(self, req, template, data, content_type`, `process_request(self, req)`, and `_build_graph(self, req, tkt_id)`.

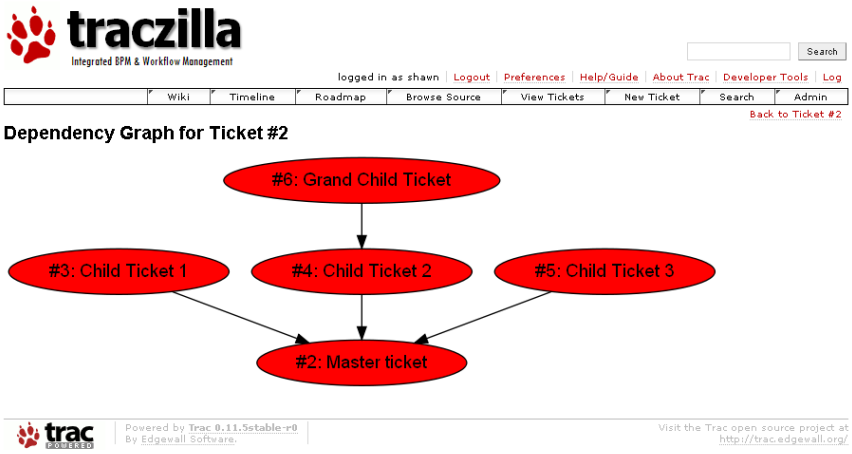

Figure 6.2: TicketDependencyPlugin Screenshot.

6.2.4 WhiteboardPlugin

The Whiteboard plugin is a 3rd party plugin which provides a shared screen in the form of a Virtual Whiteboard. It can be used by Knowledge workers to drag-n-drop tickets across statuses, team members, business requirements and user stories (ad hoc business processes) - i.e. ticket "triage" activities. See screenshot below in Figure 6.3:

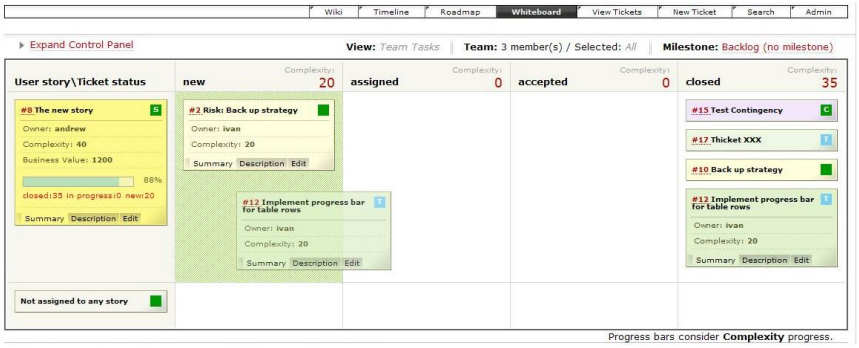

Figure 6.3: WhiteboardPlugin Screenshot.

6.2.5 StatisticsPlugin

The Statistics module is a modified 3rd party plugin designed to estimate recent user activity. It can be used by Management to generate useful reporting charts gauging: Wiki/Ticket/SVN activity. This plugin utilises the Open source FlashChart utility, for generating the line and pie graphs.

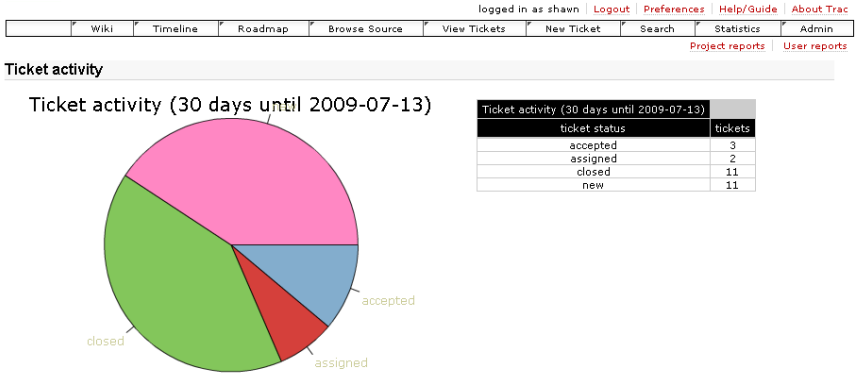

Figure 6.4: StatisticsPlugin Screenshot.

6.2.6 CustomFieldPlugin

This plugin provides a Web Admin panel for administrating custom fields - adding, modifying and deleting them without editing the trac.ini configuration file directly. It extends the TicketSystem model by adding the class `CustomFields(Component)` with methods to get, create, verify, update, and delete: `_custom_field(self, env, customfield)`.

6.2.7 MultipleWorkflowPlugin

This MultipleWorkflowPlugin replaces the ConfigurableTicketPlugin used by Trac to control what actions a ticket can do. With this plugin, Traczilla can define a workflow based on the type of ticket. Thus, customised workflow can be setup for certain types of issues/tickets/business processes. However, if there is no workflow defined for the ticket type in question, than Traczilla will use the default workflow. This plugin builds on the trac.ticket API and implements the following ITicketActionController methods:

`get_ticket_actions(self, req, ticket):` Returns a list of (weight, action) tuples that are valid for this request and this ticket.

`get_all_status(self):` Return a list of all states described by the Ticket workflow.

`render_ticket_action_control(self, req, ticket, action):` Render the valid ticket actions based on the Ticket types workflow.

`get_ticket_changes(self, req, ticket, action):` Enumerate all ticket changes.

`_has_perms_for_action(self, req, action, resource):` Verify that the user has the appropriate permissions to perform the required action.

6.3 Test Cases

In the following section, we focus our testing effort on the core Traczilla functionality. That is, on its ability to manage and support "dynamic business processes" - i.e. Collaborative and Ad hoc business processes (as explained in Chapter 2). The activity diagrams below visualise the potential usage patterns for Traczilla. In addition, unit tests are provided herewith, to verify correct underlying plug-in functionality.

6.3.1 UC-001: Collaborative Business Processes

Figure 6.5 below, shows the UML Activity diagram for UC-001. The paths that can be followed integrate elements from UC-004 (Process support) and UC-005 (Management reporting), to provide a more realistic and complete usage scenario.

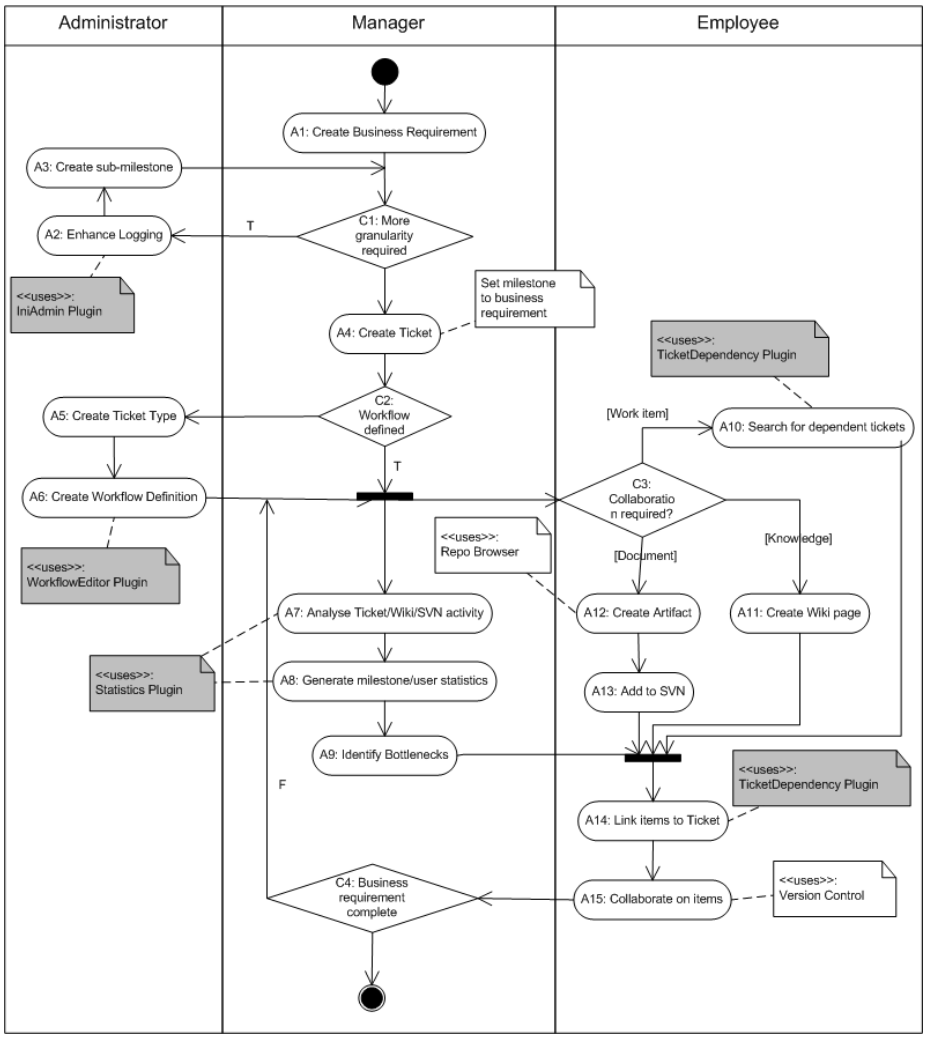

Figure 6.5: Collaborative UML Activity Diagram

Paths and Tests

Purpose		To test that businesses can use Traczilla for Collaborative Business Processes		
Use Case/s		UC-001, UC-004, UC-005		
Pre-condition		Administrator has TRAC_ADMIN, Manager has WORKFLOW_ADMIN, and Employee has TICKET_ADMIN rights		
#	**Path**	**Description**	**Component/s**	**Outcome**
1	[A1, C1, A4, C2, A7, C3, A12, A8, A13, A9, A14, A15, C4, END]	Document collaboration for a business requirement with a predefined process and simultaneous management reporting.	- Repo Browswer - *Statistics Plugin*	PASS
2	[A1, C1, A2, A3, C1, A3, C1, A4, C2, A5, A6, C3, A11, A14, A15, C4, END]	Wiki collaboration for an operational business process (more detail) defined at run-time.	- Wiki Module - *WorkflowEditor*	PASS
3	[A1, C1, A4, C3, A10, A14, A15, C4, END]	Collaboration involving inter-dependant work items, allocated at run-time.	- *TicketDependency Plugin*	PASS
4	[A1, C1, A4, C2, C3, A12, A13, A14, A15, C4, A7, A8, A9, C3, A10, A14, A15, C4, END]	Document and wiki collaboration involving inter-dependant work items, and simultaneous management reporting.	- Repo Browser - *Statistics Plugin* - Wiki Module - *Ticket Dependency*	PASS

IniAdmin Unit Tests

ID	**Input**	**Purpose**	**Expected Output**	**Actual Output**
IA-1	LOG_LEVEL: debug	Legal test case	Confirmation of change, ini file updated.	As expected
IA-2	LOG_LEVEL: asdghr	Invalid entry	Error shown, ini file not updated	FAIL (file updated)
IA-3	LOG_LEVEL set through File and WEB UI simultaneously	Race case	In-determinant	As expected

Statistics Plugin Unit Tests

ID	Input	Purpose	Expected Output	Actual Output
SP-1	Default settings	Legal test case	Generate 3 month activity report	As expected
SP-2	Weeks Back = 'abc'	Invalid entry	Error shown	PASS (input ignored)
SP-3	Default User Report	Legal test case	Generate charts for the most active user	As expected

TicketDependency Plugin Unit Tests

ID	Input	Purpose	Expected Output	Actual Output
TD-1	Refers To: '#1', '#3'	Legal test case	Dependency graph with all 3 nodes shaded red (in progress)	As expected
TD-2	Refers To = 'abc'	Invalid entry	Should not be possible to reach this state	As expected
TD-3	Refers To: '#1' [completed]	Boundary case	Dependency graph shows relationship with ticket #1 (related ticket) in green	As expected
DS-1	Search = 'workflow'	Legal test case	Return results	As expected
DS-2	Search = "' SELECT * FROM permission"	Illegal entry (SQL injection)	No results shown	As expected
DS-3	Search = ''	Boundary case	Return error	PASS (No results)

6.3.2 UC-002: Ad hoc Business Processes

Figure 6.6 below, shows the UML Activity diagram for UC-002. To provide a more realistic and complete usage scenario, the paths that can be followed also integrate elements from UC-002 (Work item support), UC-004 (Process support) and UC-006 (Customisation).

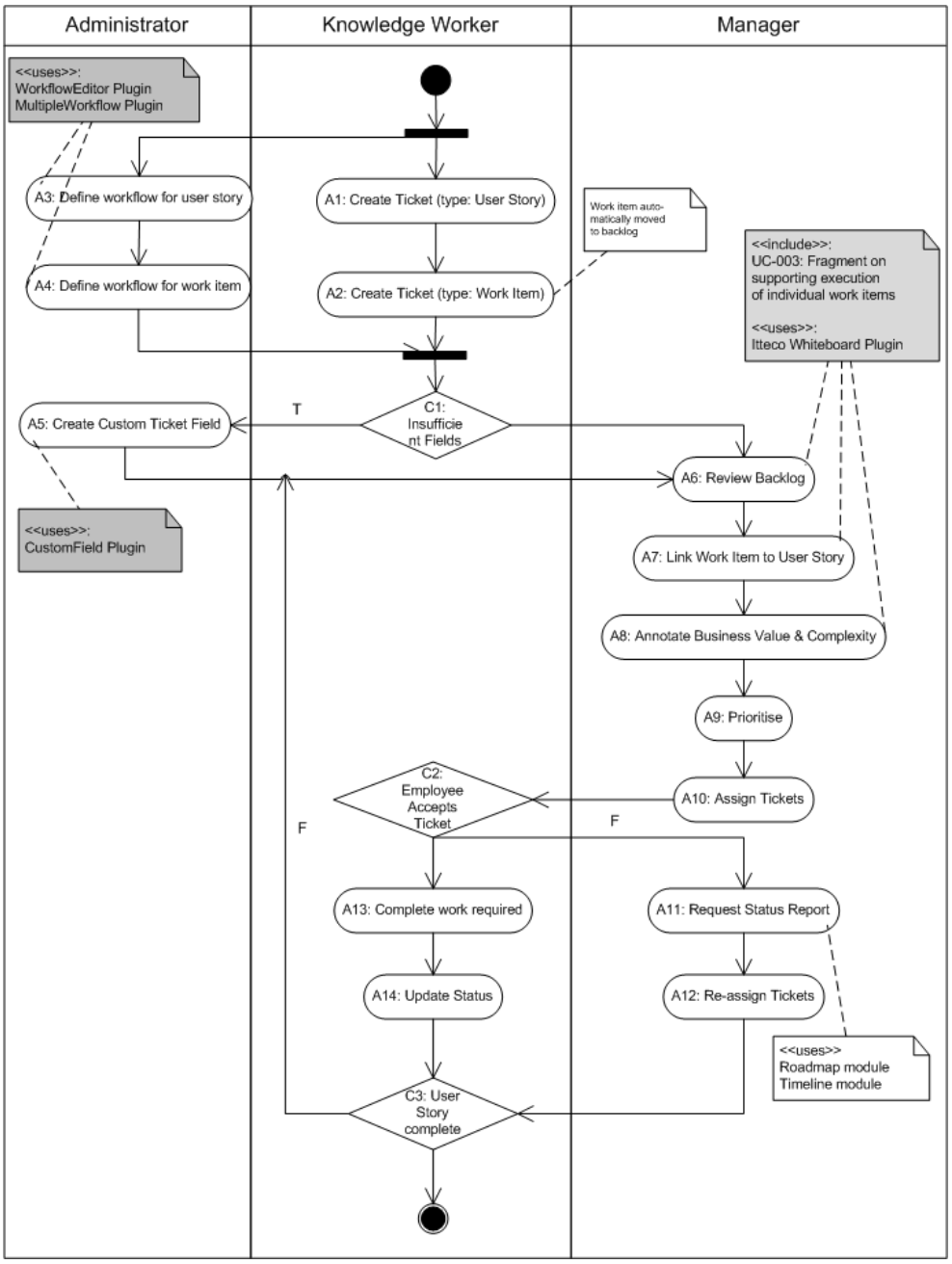

Figure 6.6: Ad hoc UML Activity Diagram

Paths and Tests

Purpose		To test that businesses can use Traczilla for Ad hoc Business Processes		
Use Case/s		UC-002, UC-003, UC-004, UC-006		
Pre-condition		Administrator has TRAC_ADMIN, Manager has WORKFLOW_ADMIN, and Employee has TICKET_ADMIN rights		
#	**Path**	**Description**	**Component/s**	**Outcome**
5	[A1, A2, C1, C2, A13, A14, C3, END]	Individual (administrative) work items defined at run-time, with all work assigned through ticket system.	- Ticket System	PASS
6	[A1, A3, A2, A4, C1, A6, A7, A8, A9, A10, C2, A13, A14, C3, END]	Ad hoc process defined at run-time, utilising the Whiteboard to annotate and assign tickets.	- Ticket System - *WorkflowEditor* - *MultiWorkflow* - Whiteboard	PASS
7	[A1, A2, C1, A5, C2, A11, A12, C3, C2, A13, A14, C3, END]	Ad hoc process customised with additional ticket fields, also involving management reassignment of tickets.	- Ticket System - *CustomField plugin*	PASS
8	[A1, A2, C1, [[A6-A10]], C2, A13, A14, C3, , [[A6-A10]], C2, A13, A14, C3, END]	Ad hoc process involving several iterations, using Whiteboard to monitor and reallocate issues at run-time.	- Ticket System - *Whiteboard*	PASS

WorkflowEditor Unit Tests

ID	**Input**	**Purpose**	**Expected Output**	**Actual Output**
WF-1	Create new ticket, TYPE = "Task" (workflow already defined)	Legal case	Use 'Task' workflow definition.	As expected
WF-2	Create ticket, TYPE = "Enhancement" (no workflow defined)	Boundary case	Use default workflow definition	As expected
WF-3	Set DEFAULT workflow definition to: NULL	Illegal case	Display error	PASS (Reject input)
WF-4	Set TASK workflow definition to: NULL	Boundary case	Revert to default workflow definition	FAIL (Input rejected)
WF-5	Set DEFAULT workflow definition to: "asdfasd; asdfjk dsf"	Illegal case	Display error	PASS (Reject input)

WF-6	Set TASK workflow definition to: "close = new - ¿ closed"	Legal case	Update workflow definition	As expected
WF-7	View graph for created workflow definition	Legal case	Workflow graph shown in GraphViz notation	FAIL (GraphViz timeout)

CustomField Plugin Unit Tests

ID	Input	Purpose	Expected Output	Actual Output
CF-1	Create new field	Legal test case	Field added to all tickets	As expected
CF-2	Create existing field	Illegal test case	Error shown, field not added	As expected
CF-3	Remove custom field	Legal test case	Field removed from ticket display (underlying data kept)	As expected

6.3.3 Scenario

A real world scenario in which Traczilla would be a suitable tool to manage a *dyanmic* business process, would be in an organisation where there is a need to develop an annual budget (e.g. a Hotel company). Developing an annual budget requires a number of parallel and collaborative tasks, including: historical data collection; financial projections; authorisation of projections; and manipulation of artifacts (such as Budget spreadsheets). In this scenario, one could use Traczilla by firstly creating a Milestone: "Develop Annual budget" (this way all activities relating to the budget can be aggregated in the formation of management reports). Secondly, managers would then assign a ticket (type: user story) to an employee/team titled: "Collect historical data". One employee would then take ownership of this user story and delegate specific tickets (type: task), such as: "Balance sheet figures", "Income statement figures", and "Cashflow statements".

After delegation of tickets has occurred, the employees who have been assigned tickets, would update the status of the ticket as work is done, and add any artifacts (such as spreadsheets), to the Repo browser (via SVN), and then link these artifacts to the ticket. By adding these artifacts through SVN, it also allows other employees who need to contribute to the the spreadsheet to manipulate the file at the same time, and commit their changes, without waiting on another employee. A similar process would occur for "Financial projections".

The "Authorisation of projections", would require delegating tickets to management for authorisation or confirmation that the Budget figures are suitable. One of the major benefits to utilising Traczilla in this manner is that, once the Annual budget is complete, management can get detailed statistics on which users performed which tasks, and where the biggest bottlenecks were in the process. Thus, allowing improvement during the development of next years annual budgets.

6.4 WorkflowEditor Analysis

Listing from source file `workfloweditor.py`

```
# -*- coding: utf-8 -*-
# Created by Shawn McLaren on 2009-06-24.
# Copyright (c) 2009 Shawn McLaren. All rights reserved.

from trac.core import *
from trac.web.chrome import ITemplateProvider, add_stylesheet, add_script
from trac.admin import IAdminPanelProvider
from trac.web.api import ITemplateStreamFilter, IRequestHandler
from trac.web.chrome import Chrome
from trac.web.href import Href

class WorkflowEditorAdmin(Component):
    implements(ITemplateProvider, ITemplateStreamFilter, IAdminPanelProvider

    # ITemplateProvider method
    def get_htdocs_dirs(self):
        from pkg_resources import resource_filename
        return [('workfloweditor', resource_filename(__name__, 'htdocs'))]

    # ITemplateProvider method
    def get_templates_dirs(self):
        from pkg_resources import resource_filename
        return [resource_filename(__name__, 'templates')]


    # ITemplateStreamFilter method
    def filter_stream(self, req, method, filename, stream, data):
        return stream

    # IAdminPanelProvider method
    def render_admin_panel(self, req, cat, page, path_info):
        req.perm.assert_permission('TRAC_ADMIN')
        add_script(req, 'workfloweditor/js/jquery.jqGrid.js')
        add_script(req, 'workfloweditor/js/grid/jqModal.js')
        add_script(req, 'workfloweditor/js/grid/jqDnR.js')
        add_script(req, 'workfloweditor/js/grid/jquery.tablednd.js'
        add_script(req, 'workfloweditor/js/ui/ui.core.js')
        add_script(req, 'workfloweditor/js/ui/ui.tabs.pack.js')
        add_script(req, 'workfloweditor/js/workfloweditor.js')
        add_stylesheet(req, 'workfloweditor/css/grid.css')
```

```
        add_stylesheet(req, 'workfloweditor/css/jqModal.css')
        add_stylesheet(req, 'workfloweditor/css/ui.tabs.css')
        add_stylesheet(req, 'workfloweditor/css/workfloweditor.css'

        # Determine the type of workflow
        png = ''
        url_path = req.path_info.split('/')
        if req.path_info.endswith('workfloweditor'):
            type=''
        else:
            type = url_path[3]

        if type in ('enhancement', 'defect', 'task', 'testing'):
            page_template = 'workfloweditor_admin-%s.html' % type
        else:
            page_template = 'workfloweditor_admin.html'

        if req.method == 'POST':
            self._update_config(req,type)

        page_param = {}
        self._create_page_param(req, page_param, type)

        return page_template, {'template': page_param}

class WorkflowChangeHandler(Component):
    implements(IRequestHandler)

    # IRequestHandler method
    def match_request(self, req):
        match = False
        if req.path_info.startswith('/admin/ticket/workfloweditor')
            match = True

        return match

    # IRequestHandler method
    def process_request(self, req):
        req.send_response(200)
        req.send_header('Content-Type', 'content=text/html; charset=UTF-8')
        req.end_headers()
        req.write("OK")
```

Listings

The fragment above, shows the code necessary to render the WorkflowEditor in the Admin Panel, as shown below in Figure 6.7.

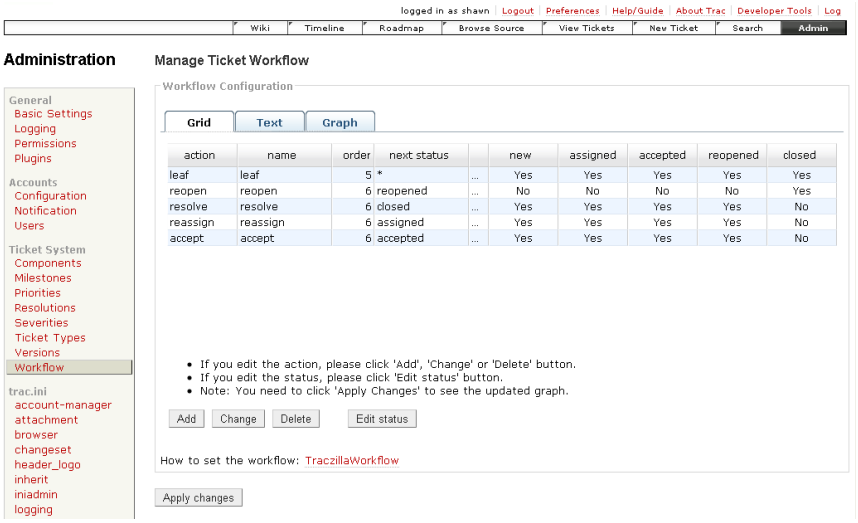

Figure 6.7: WorkflowEditor Screenshot

This workflow definition can be edited in the Grid view, providing a user friendly interface for `rawactions` defined by trac. The syntax for these workflow definitions involves the following operations:

- del_owner: Clear the owner field.
- set_owner: Sets the owner to the selected or entered owner.
- set_owner_to_self: Sets the owner to the logged in user.
- del_resolution: Clears the resolution field
- set_resolution: Sets the resolution to the selected value.
- leave_status: Displays "leave as current status" and makes no change to the ticket.

Example:

```
resolve_new = new -> closed
resolve_new.name = resolve
resolve_new.operations = set_resolution
resolve_new.permissions = TICKET_MODIFY
resolve_new.set_resolution = invalid,wontfix
```

There are a couple of hard-coded constraints to the workflow. In particular, tickets are created with status new by default, and tickets are assumed to have a closed state. Furthermore, the default reports/queries treat any state other than closed as an open state. Shown below in Figure 6.8 is the default workflow model used (in graph notation), if no other definition is provided

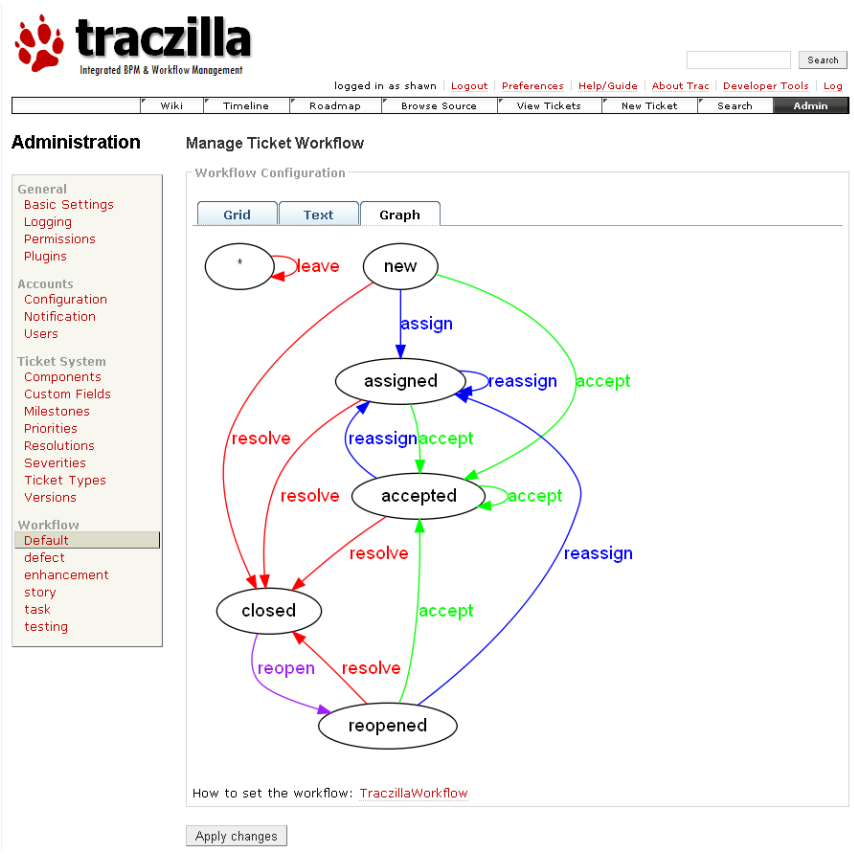

Figure 6.8: Default workflow definition.

6.5 Summary

In summary, a total of 30 test cases were executed on the Traczilla system. Of these 30 test cases, 3 defects were found and 21 test cases were closed off immediately with no bugs found. This is equivalent to a 10% defect rate. However, the defects found did not affect the execution of the use cases, and as such the core functionality that Traczilla was designed for is unaffected by these defects. Most of the defects found relate to boundary cases or illegal cases. For instance, the first defect found in the IniAdmin plugin (IA-2), was because user input was not validated before updating the ini configuration states, leading to the possibility of illegal states. However, the effort required to remedy this bug did not appear to be worth the benefit.

The second bug was found in the WorkflowEditor plugin (WF-4), whereby it was not possible to delete an existing workflow definition for a ticket type. This is is a boundary case, because the only time a user would need this is if they would need to revert to the Default workflow. Nonetheless a workaround is still possible, whereby the user can manually delete the workflow definition from the Ini file, and thereby revert to the default workflow definition.

The final bug found was also in the WorkflowEditor plugin (WF-7), whereby the graph component would display intermittant behaviour. That is, some-times working, some-times not, a diagnosis revealed that the root cause of the problem was a timeout occurring in the 3rd party utility - GraphViz. No workaround could be found to remedy this problem. Despite these bugs, the underlying Traczilla functionality is still supported, and the core features provided in each plugin work as intended.

Part V

Summary

Chapter 7

Conclusion

In this final chapter, we summarise our main findings, provide answers to our initial research questions, and any added insight gained throughout the research, design, and development phases. We also highlight any implications emerging from the research and development presented herewith, and end by discussing potential future research avenues.

7.1 Main Findings

The main findings of this Thesis, relate back to the initial research questions, presented in chapter 1. We provide these research questions below, as well as the answers we have derived.

7.1.1 Research Question I

Why is current BPM Technology inadequate for managing dynamic business processes?

As stated in chapter 2, the main goals of BPM are traceability, flexibility, and a better understanding of operations. This understanding and flexibility can be easily supported in the case of static, well structured business processes. However, these goals are far more difficult to support when trying to manage dynamic business processes.

The four main categories of BPM technology are: CSCW, Production WfMS, Ad hoc WfMS, and CHS. Each technology approaches the problem with different sets of tools implemented from different perspectives. Each manages the dependencies between activities differently in addition to making different philosophical assumptions about how the system should be used to accomplish work. However, usage has shown that no single system brings together all the features needed to address the level of flexibility and evolution required to manage dynamic, real world business processes [7].

What characteristics embody dynamic business processes?

To answer this question, we return to our classification of business processes presented in chapter 2. Here it was shown that a process may be system-centric, (i.e. performed automatically by an application); or human centric (i.e., manual tasks involving human judgment, or manual processing of documents). In addition, a process can be classified as: production/administrative (i.e. highly-structured); ad hoc (i.e. semi-structured); or collaborative, where there exists no repeatable patterns or sequences among the tasks, and participants often need to collaborate to perform the work (i.e. unstructured).

These classifications lead to the conclusion that dynamic business processes can be characterised as in-frequent and ad hoc, involving unstructured human collaboration.

What problems exist when handling dynamic workflow in current BPM Tools?

CSCW is a technology that is primarily designed with flexibility in mind, and as such there is at times insufficient structure to maintain clarity and awareness of the underlying business process. Traditional WfMSs support static workflows, and minor exceptions that cause variation from pre-defined models well, but this support is not scalable and flexibility is still deemed insufficient. Ad hoc WfMSs and CHSs, provide a good balance in providing support and flexibility. However, while these systems represent a promising approach to coordination, they have yet to be widely adopted.

What are the limitations regarding Email communication?

Email has been the emergent solution to this thorny problem of handling dynamic workflow. As such, email has evolved from a mere communication system to a means of organizing workflow, storing information and tracking tasks. However, the vast increase in email volume and use of email as a multi-functional tool now threatens the productivity gains once created. As a result, professionals now spend an uncomfortable amount of time simply organizing and storing email; creating problems for emails involving critical business processes.

What makes SCM Tools suitable for managing dynamic business processes?

Support for dynamic change in a workflow infrastructure, requires the ability to dynamically change a process's definition, and the execution model at run-time. This kind of functionality is built-in to many modern day SCM tools implicitly. Thus adopting such technology would allow for better adaptation to changing requirements. In addition, the convenience of artifact tracking and hand-off tools provided by SCM technology, allows complex, multi-person, ongoing workflow, to be supported without defining all task requirements up-front.

Furthermore, the synonymous goals: flexibility, transparency and control (i.e. structure). The fact that these tools are used in distributed locations, and that they can provide traceability from business request to implementation - also support this premise.

7.1.2 Research Question II

What are the design criteria for a system supporting dynamic business processes?

What functionality is available within current SCM Tools?

Traditional CM is largely focused on change control procedures associated with evaluation, co-ordination, approval, and implementation of changes. Good status accounting is another element, providing data capture, data recording, and report generation - useful for gauging performance characteristics, fine tuning estimation, or identifying bottlenecks. Lastly, auditing is provided to verify that all issues are solved, and that the proper process has been followed and proper activities have been applied in solving a particular issue.

This functionality has been best enabled through modern-day developer-oriented SCM tools, which moved the focus towards: workspace management, version control, and generic process support. Further advancements were made with the invent of Agile methods, and the Issue Tracking tools that ensued. In the Trac system for example, users can use the ticket system to assign tasks, make comments and to discuss issues. This makes understanding the motivation behind a decision- or implementation choice easier, when returning to it later.

What additional functionality is needed to manage dynamic business processes?

A more balanced trade-off between flexibility and support was the main functionality, which came in the form of the WorkflowEditor plugin and the TicketDependency plugin. In addition, functionality was needed for facilitating human co-ordination and collaboration, which came in the form of integrating a Wiki, and a Virtual Whiteboard.

Different levels of business process also needed to be identifiable, ranging from high-level business strategies describing long-term strategic goals, to implemented business processes prescribing the execution of process activities. This was implemented by adding a milestone hierarchy to Traczilla.

What are the other design decisions regarding UI design, Database design and Back-end architecture?

A number of these design decisions were made after the decision to base all development on the fully functional *Trac* Issue Tracking solution. This decision was motivated by two main factors: 1) Trac is a tried and tested solution, offering a higher degree of confidence in its functionality; and 2) the Back-end component architecture of Trac is easily extended via plugins.

Thereafter, the design decisions regarding the UI design, involved accepting the knowledge worker as an important source, and allowing total configuration of the Traczilla system. The database design was largely based on the Trac database model with only minor changes needed to make Traczilla more applicable to business users.

7.1.3 Research Question III

What are the functional and non-functional requirements for the Traczilla System?

A summary of the core requirements and Traczilla implementations is given below:

Design Trait	**Traczilla Implementation**
Run-time dynamism	*WorkflowEditor*, supports binding ticket behaviors and process elements at run time. Allows on-the-fly composition and change of workflow definitions.
Configurable execution	*MultipleWorkflowPlugin*, allows execution model to be tailored to individual workflow needs. *WhiteboardPlugin*, allows execution to be reflexively controlled by human participants.
Limits Legal Liability	Timeline module & *TicketDependencyPlugin*, allows for traceability from business request to implementation. *StatisticsPlugin*, visualises reporting data allowing enhanced metrics, and measurements for detecting bottlenecks etc.
Logically decomposable models	*IniAdminPlugin*, allows process to be hierarchically decomposed into sub-milestones, providing abstraction and separation of responsibilities and ownership.
Reusable fragments/ components	Development of generalised plugins (e.g. *CustomFieldPlugin*), with customisable field definitions and coding standards. N.B: Traczilla plugins may be manipulated and reused in other components.
Adoption/ Integration	Highly componentized open system, supporting adoption in pieces and integration with tools in different fields. Views may be tailored to individual participants. Use of standard protocols and tools for distribution limits buy-in cost (adoption).
Support for participant communication	Integration with third party CSCW tools: whiteboard, wiki, and workflow technology.

Table 7.1: Traczilla design considerations and implementations.

7.2 Implications of Research

The main findings of this research, highlight three key points. Firstly, managing dynamic business processes with current BPM technology (particularly email) is currently ineffective. Secondly, the main reason behind this ineffectiveness is that a successful integration of BPM tools has currently not been found. Meaning, lastly, there is a definite potential to apply SCM tools in the BPM field, as they provide an integration of tools that would allow businesses to push for further agility - as motivated by the Agile manifesto.

If successful, the implications for this research could be very important for businesses, as it would represent a new way of communicating and executing work flows for both small and large companies alike. It would also lower the importance of email communication, and help businesses to better track issues so that things do not "slip through the cracks".

7.3 Further Research

There are numerous future research avenues that this study reveals. First and foremost, is that of simulating a Traczilla case study, in order to gauge exactly how useful this technology could be for businesses, and also to identify any weak design points that may need improvement. Gathering information on whether the right balance has been made between flexibility and structure is also critical. As, in the words of Kammer [23]: "the most successful technologies are ones where a balance is maintained between unobtrusive work models and lightweight, lowcost of adoption and usage versus structured, managed work and reconfigurable collaboration technologies."

Furthermore, additional research is needed on what is required to enable businesses to adopt the Traczilla technology. One of the major stumbling blocks is clearly moving organisations away from email communication, which is currently one of the staple components of any businesses infrastructure. This opens up further research into examining what kind of preparation and planning is required to enable a successful Traczilla implementation. At the top of the list is clearly obtaining management support for such a system. However, there are other barriers to over come as well, such as: technical, cultural, political, risk-related etc. All of these areas require further examination to determine the exact market model for Traczilla.

Lastly, the area of business intelligence, is another which offers significant potential for future Traczilla implementations. Investigation and development in the areas of Process Mining, Statistical Analysis, and OLAP are key to expanding the market usability and suitability of Traczilla in the workplace.

Part VI

Appendix

Appendix A

BPM Semantics

In this appendix item, we expand on a number of semantic details concerning Business Process Management. This is included for the interested reading seeking further clarity on certain aspects of BPM.

A.1 Business Process Lifecycle

The complete BPM lifecycle is shown in figure A.1.

A.1.1 Design

Process Design encompasses both the identification & review of existing processes and the design & validation of "to-be" processes [39]. Areas of focus include: representation of the process flow, the actors within it, alerts & notifications, escalations, Standard Operating Procedures, Service Level Agreements, and task hand-over mechanisms. The purpose of the process design phase is the identification of those processes an organization wishes to analyze, redesign, and/or automate [42]. Good design reduces the number of problems over the lifetime of the process.

A.1.2 Modeling

Modeling (and implementation) takes the theoretical design and firstly maps them using (semi-)formal modeling methods. Thereafter, simulation techniques are utilised by introducing combinations of variables, for instance, changes in the cost of materials or increased rent, that determine how the process might operate under different circumstances [1].

[1] It also involves running "what-if analysis" on the processes: e.g. "What if I have 75% of resources to do the same task?" or "What if I want to do the same job for 80% of the current cost?"

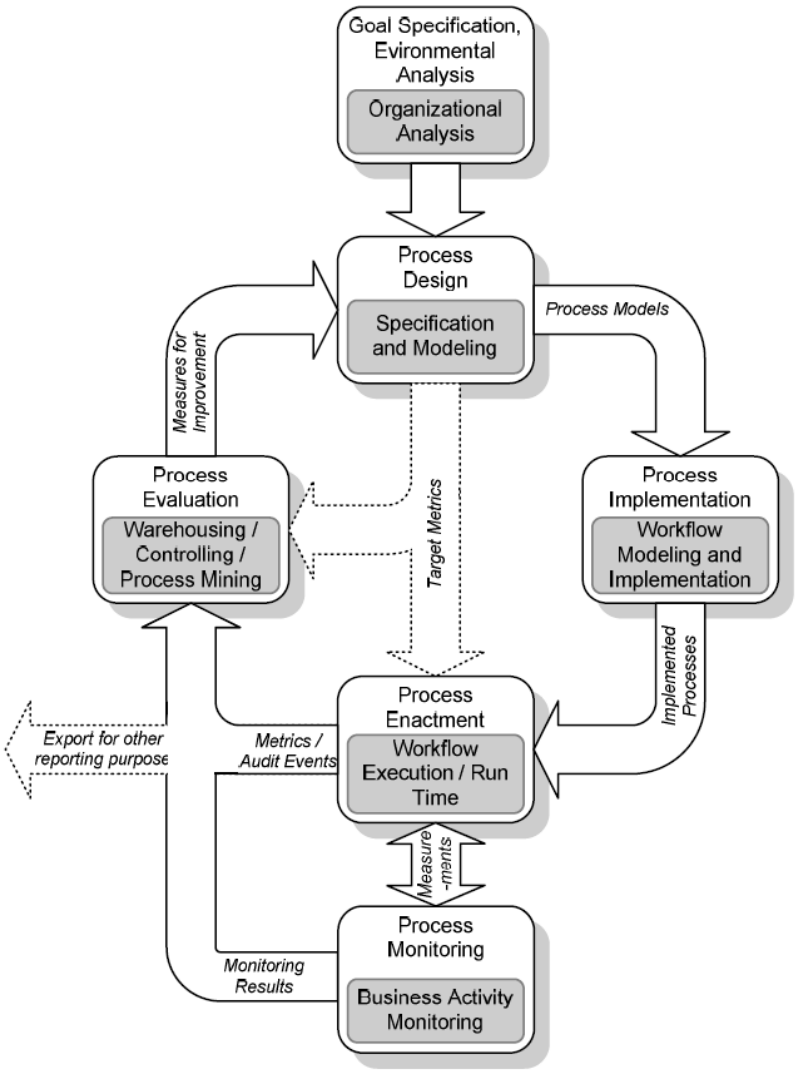

Figure A.1: Business process management lifecycle, [42]

A.1.3 Enactment

During the process enactment phase the specified process models are transferred into the operational environments which can either be manual (e.g. via procedure handbooks) or automated (e.g. via BPM or workflow software) [42]. Process enactment encompasses the run-time behaviour of business processes, such that instances are initiated at the correct times to fulfil the business goals of a company. Process enactment ensures that all process activities are performed according to the execution constraints specified [39].

During business process enactment valuable execution data can also be gathered. One of the ways to subsequently automate processes is to implement an application that executes the required steps of the process. However, in practice, these applications rarely execute all the steps of the process accurately or completely. Another approach is to use a combination

of software and human intervention; however this approach is more complex, making the documentation process difficult.

As a response to these problems, software has been developed that enables the full business process (as developed in the process design activity) to be defined in a computer language which can be directly executed by the computer. The system will either use services in connected applications to perform business operations (e.g. calculating a repayment plan for a loan) or, when a step is too complex to automate, will ask for human input. Alternatively, business rules can then be used by systems to provide definitions for governing behavior, and a business rule engine can be used to drive process execution and resolution. Compared to either of the previous approaches, directly executing a process definition can be more straightforward and therefore easier to improve. However, automating a process definition requires flexible and comprehensive infrastructure, which typically rules out implementing these systems in a legacy IT environment.

A.1.4 Monitoring

Monitoring encompasses the tracking of individual processes, so that information on their state can be easily seen, and statistics on the performance of one or more processes can be provided. Monitoring is necessary in order to visualise the status of business process instances [39]. An example of the tracking is being able to determine the state of a customer order (e.g. ordered arrived, awaiting delivery, invoice paid) so that problems in its operation can be identified and corrected. Examples of the statistics are the generation of measures on how quickly a customer order is processed or how many orders were processed in the last month. These measures tend to fit into three categories: cycle time, defect rate and productivity. The degree of monitoring depends on what information the business wants to evaluate and analyze and how business wants it to be monitored, in real-time or ad-hoc.

A.1.5 Evaluation

During process evaluation, the audit trails produced during the process enactment and monitoring stages are used for the purpose of process control and improvement. During this stage data from multiple process instances are aggregated to discover temporal trends and design flaws. Feedbacks and contingency plans for process improvement can be formulated based on the results of process measurement and evaluation. [42].

Information extracted from the monitoring phase is evaluated using business activity monitoring and process mining. Business activity monitoring (BAM) is used to identify activities that take too long due to a shortage of resources. Process mining is a collection of methods and tools related to process monitoring. The aim of process mining is to analyze event logs extracted through process monitoring and to compare them with an 'a priori' process model. Process mining allows process analysts to detect discrepancies between the actual process execution and the a priori model as well as to analyze bottlenecks [36].

Ultimately, process evaluation boils down to: retrieving process performance information from modeling or monitoring phase; identifying the potential or actual bottlenecks

and the potential opportunities for cost savings or other improvements; and then, applying those enhancements in the design of the process. Overall, this creates greater business value.

A.2 Evolution of Workflow Tools

Many types of product in the IT market have supported aspects of workflow functionality for a number of years, yet it is only comparatively recently that its importance has been recognised in its own right. The evolution of workflow as a technology has thus encompassed a number of different product areas. The following section present a wide range of the familiar technology that has contributed to the development of modern-day workflow systems [20].

A.2.1 Image Processing

Workflow has been closely associated with image systems for many years and nowadays many image systems have workflow capability either built-in or supplied in conjunction with a specific workflow product. Today there are countless business procedures that involve interaction with paper-based information, which may need to be captured as image data as part of an automation process. Once paper based information has been captured electronically as image data, it is often required to be passed between a number of different participants for different purposes within the process, possibly involving interaction with other IT applications, thereby creating a requirement for workflow functionality. Image processing systems were thus one of the early success stories of workflow management technology, and have been used to great effect in improving the efficiency of many *production* workflows.

A.2.2 Document Management

Document management technology was one the first driving forces behind document-centric workflows (as discussed above). This technology is concerned with managing the lifecycle of electronic documents, and includes facilities for: managing document repositories distributed within an organization, routing documents (or even separate parts of documents) to individuals or; updating documents according to their specific roles relating to a specific document. Given that a document may form part of a particular business procedure, "document-centric" workflow technology have proved useful in enhancing the efficiency of many *administrative* workflows.

A.2.3 Electronic Mail

Email is an integral tool for many of todays organisations, and provides powerful facilities for distributing information between individuals within an organisation or between

organisations. Thus, email systems have themselves been progressing towards workflow functionality through the addition of routing commands to define a sequence of recipients for particular types of mail items in response to some form of identified business procedure. In the next chapter we return to these "mail-centric" systems, and illustrate the limitations and drawbacks of utilising email for workflow and task management purposes.

A.2.4 Database Applications

From the early days of databases, emerged support for certain classes of business procedures, i.e. "transactions". From organisations initial centralised style of working, database application software has increasingly enabled the distribution of transaction based applications across a number of computer platforms. Transaction based applications typically exhibit important characteristics of robustness and support for "atomic" properties of the transaction; however, they do not typically exhibit a separation between the business procedure logic and the invocation of the various application tools which may be required to support individual activities within the business process. Over time, this is leading to a requirement to consolidate workflow capabilities to control the business procedures with the ability to invoke traditional transaction application programs for appropriate parts of the business process. Nonetheless database applications are still one of the firm building blocks of workflow evolution.

A.2.5 Project Support Software

As discussed in the previous chapter, project support software can be used to handle complex IT applications and project development. More recently, this technology has started to provide a form of workflow functionality within the project environment, for "transferring" development tasks between individuals and routing information between individuals to support these tasks. In any case this type of software can still be generalised to support a wider, business-oriented view of processes and a wider range of application tools - offering a more general workflow capability. As such, a number of the SCM tools discussed in the previous chapter will be utilised in guiding the design of Traczilla.

A.2.6 BPR and Structured System Design Tools

Business Process Re-engineering tools have provided IT based support for the activities of analysing, modelling and (re-)defining the core business processes of an organisation and the potential effects of change in such processes or organisational roles and responsibilities associated with such processes. This may include analysis of the process structure and information flows supporting it, the roles of individuals or organisational units within the process and actions taken in response to different events, etc. A natural extension of such tools is to facilitate the implementation of the process with IT support infrastructure to control the flows of work and associated activities within the business process.

Appendix B

SCM Tools

It is an eternal truth that newly written software packages will contain bugs. To track bugs, many organizations still rely on Word documents and Excel spreadsheets, but these tactics are inefficient and error-prone to say the least. Issue tracking tools emerged as a tool which can be used to great effect in software development. As a step towards the design criteria for Traczilla, we examine Current Off-the-shelf (OTS) SCM software.

In the following sections, we look at three issue-tracking solutions: Bugzilla, Trac, and JIRA. These products were chosen, as they are among the most widely used issue-tracking tools in the development community today. Bugzilla is probably the most well-known of the open source issue-tracking solutions and is used by many high-profile open source projects such as Mozilla, Apache and Eclipse. It is a mature, feature-rich open source issue-management solution well-adapted for use in large projects. Trac is another open source issue-tracking system, but with a different approach. Trac is a lightweight, minimalistic solution, designed to allow effective issue management with as little overhead as possible. It also boasts excellent integration with Subversion. And JIRA is a well-regarded commercial product widely used in the Java community, especially among open source products.

B.1 Bugzilla

Bugzilla is one of the original Web-based general-purpose bugtracking and testing tools, and is probably the most well-known of the open source issue-management tools. It is used on many open source projects such as Mozilla, Eclipse, and many Linux distributions, and is well-adapted to large, open projects. It is designed to allow individuals or groups of developers to keep track of outstanding bugs in their product effectively. In its standard form, Bugzilla has arguably one of the ugliest and most convoluted Web sites. However, it is functional.

B.1.1 Synopsis

Bugzilla is powerful and flexible, and fits well into a multi-project environment. It allows management of multiple products, optionally grouping related projects. To assist project management and quality assurance, one can also define components, versions, development milestones and release versions. Given Bugzilla's origins as an issue-management system for open source projects, it is, by default, quite open about security: users can usually create an account themselves and create and access bugs for any project. If need be, however, it does allow one to restrict user rights to certain projects or create groups so that certain products or bugs can only be seen by certain people.

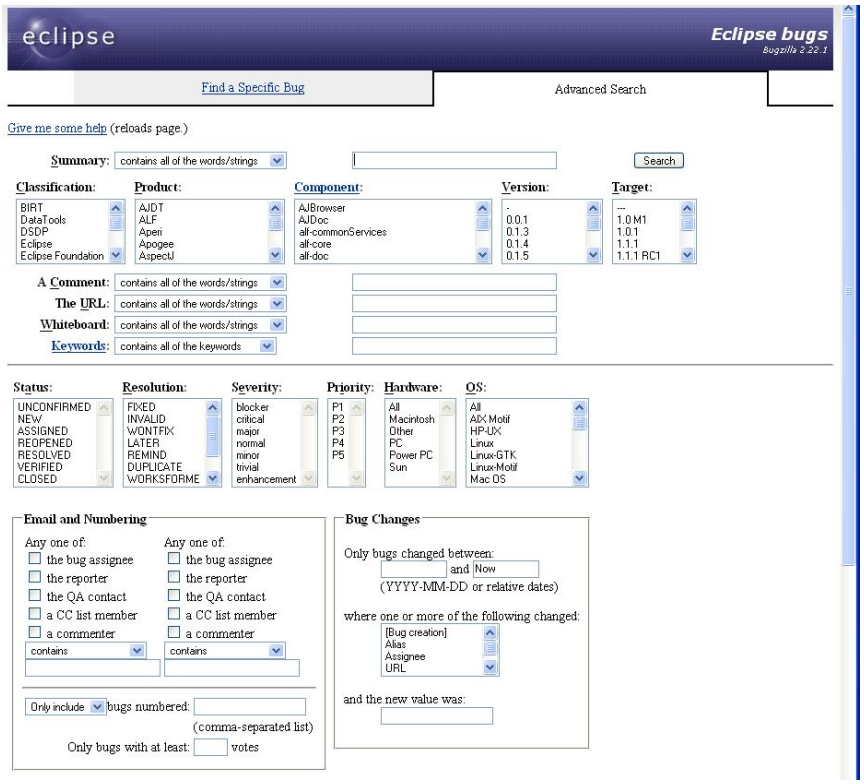

Figure B.1: The Advanced Search in the Eclipse Bugzilla database.

B.1.2 Search functionality

Given that Bugzilla is a product that manages systems containing literally hundreds of thousands of bugs, Bugzilla has powerful, if not particularly user friendly, search features. Users can search for existing issues using either a simple and convenient keyword-based search with optional filtering by product or by status, or by using the more sophisticated Advanced Search, where you can filter on virtually any field in the database (see Figure B.1).

B.1.3 Change management

Although the interface is rudimentary and lacks many of the niceties of more recent tools, entering a new issue in Bugzilla is relatively straightforward. After selecting the buggy product, the bug details screen appears. Out of the numerous fields, only the Summary and Component fields are actually mandatory. Other details can be optionally provided, such as a version number (the version of the product in which the bug was found), severity of bug, platform, a target milestone, and so on. Users can also assign a bug directly to a developer (if it is known who will be working on it), or wait for it to be picked up by some-one.

B.1.4 Workflow support

Bugzilla supports a fairly complete, albeit hard-coded workflow model (see Figure B.2). The typical life of a bug goes something like this: A tester (or user) creates a new bug. New bugs are created either as unconfirmed, new, or assigned. Typically, a developer will accept a bug (or assign it to someone else). Once the developer has corrected the bug, he or she can mark it as resolved, specifying how it was resolved: fixed, invalid (not a bug), duplicate, won't fix, and the (in)famous "works for me". A resolved bug isn't officially closed until someone from QA (quality assurance) checks it out. Once QA has confirmed the correction, the bug becomes verified. It remains in this state until the product release containing the fix actually ships, at which point, it is closed. Although this workflow model cannot be customized in Bugzilla, it usually proves sufficient for most organizations.

B.1.5 Reporting

Bugzilla provides some reporting and charting features. Users can generate tables, and simple bar or pie charts using a screen similar to the Advanced Search screen. However, to display graphs of data over time, users need to set up special "data sets" that collect the data on a regular basis.

B.1.6 Summary

Bugzilla is a powerful tool that can help a team get organized and communicate effectively. It is a tried-and-true solution that supports large projects and user bases. Its workflow features are more than sufficient for most organizations. On the downside, Bugzilla is particularly complicated to install and maintain, and the user interface wouldn't win any prizes for design or usability. Its reporting features are sufficient, though they are not particularly user friendly.

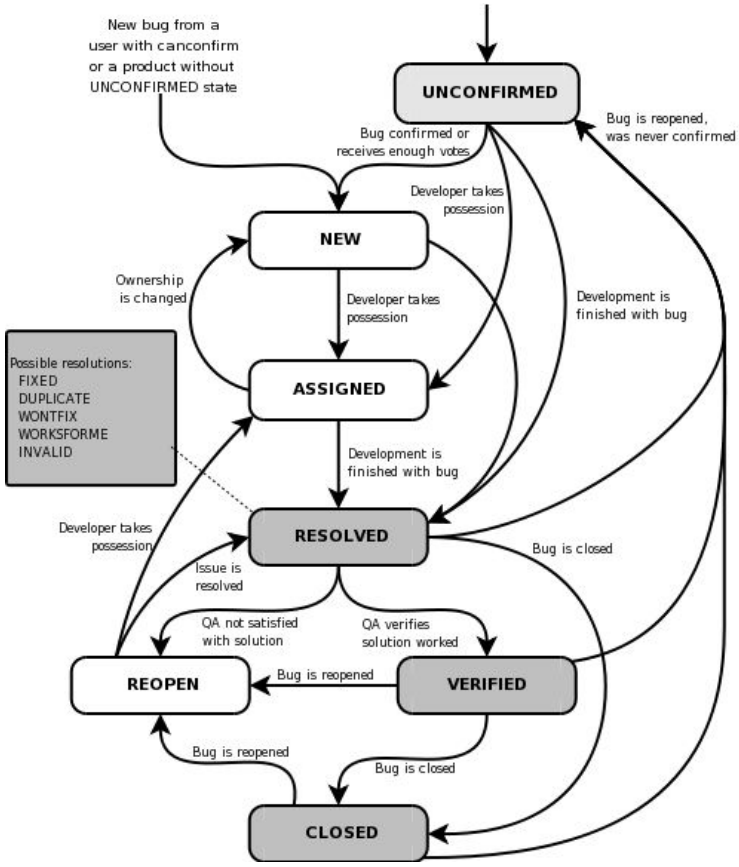

Figure B.2: The Bugzilla life cycle model, [5].

B.2 Trac

Trac is a lightweight open source, web-based project management tool that integrates a Wiki, Issue tracking, and Software Configuration Management. It is developed and maintained by Edgewall Software. Trac uses a minimalistic approach to web-based software project management, emphasising: ease of use and low ceremony. Hence, Trac is designed to impose as little as possible on a team's established development process and policies. It does not provide as many features as Bugzilla, but it is easy and intuitive to use.

B.2.1 Synopsis

Compared to other issue-management solutions, Trac takes a different approach in several areas. The first, and possibly the most striking, is that Trac is in fact a wiki[1] - meaning any page can be modified by users with appropriate rights. Wikis are a powerful tool in today's corporate world, providing vital support for knowledge management - indeed, with a minimum of communication and training, every modern organisation could probably put

[1] A Built-in documentation server

a wiki to good use. Trac is an excellent wiki for a development team, as it provides seamless integration between issue and release management, agile team communication techniques and the source code repository. This leads to the other big innovation in Trac: its close integration with Subversion[2]. This allows one to browse the source code repository directly from within Trac, displaying both the actual source code and the modification history.

Figure B.3: Example of a Trac ticket.

B.2.2 Search functionality

Trac provides a simple but powerful full-text search functionality, which lets users search not only tickets, but also wiki pages and change-sets. The search also recognizes the Trac wiki syntax, so users can go directly to a change-set, ticket or report simply by entering its number. Another way to find tickets is to use the View Tickets view, which contains many useful predefined reports, such as Active Tickets, My Tickets, or All Tickets by Milestone. It allows creation of custom queries, which can be built using an intuitive query builder.

B.2.3 Change management

The ticket system is the central element of Trac. Tickets can be used for project tasks, feature requests, bug reports, and software support issues. Trac also allows users to easily reconcile overlapping tickets (where more than one person reports the same thing). Entering a new ticket in Trac is easy (see Figure B.4). As mentioned earlier, Trac's wiki-based architecture eases the insertion of links to other tickets, change-sets, or to files in the source code repository. Once the ticket has been created, it can be viewed and updated by other users.

[2]A Version control program used in software development

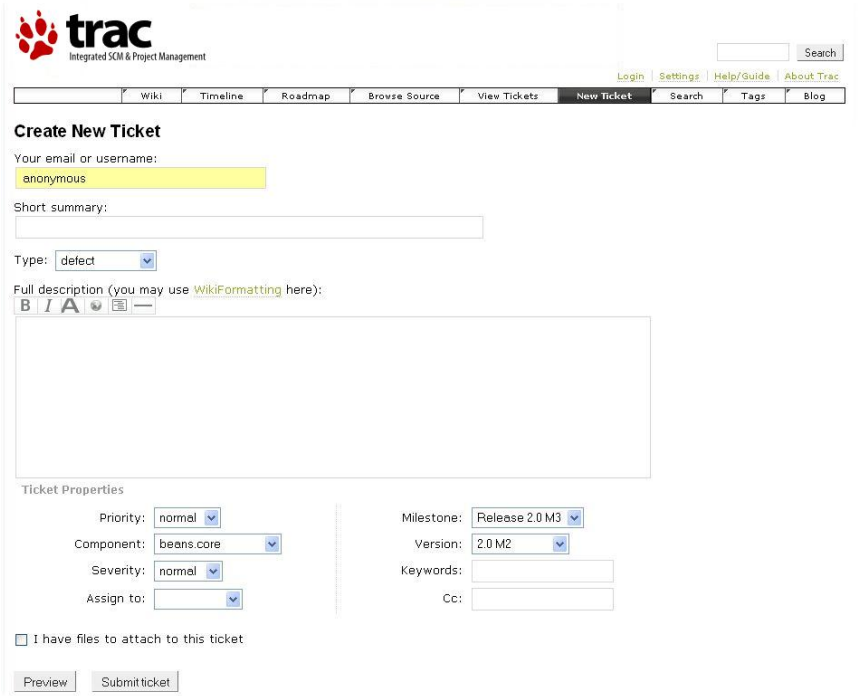

Figure B.4: Entering a new ticket in Trac.

B.2.4 Workflow support

The workflow support in Trac is lightweight. Once a ticket is created, a ticket can be assigned to (or accepted by) a user. Once it has been fixed, it is marked as "resolved". There is no provision for customised workflow, nor is there a state between "resolved" and "closed", where quality assurance can verify the correction, as found in Bugzilla and JIRA.

B.2.5 Reporting

Trac gives users a number of convenient ways to stay on top of events and changes within a project. Users can set milestones, and view a roadmap of progress towards them (as well as historical achievements) in summary. There is a timeline of individual changes so users can see the order of events, starting with the most recent. Trac also supports RSS for content syndication: allowing people to subscribe to those changes outside Trac itself.

Trac provides many views that make daily issue-management activities easier as well, so users can easily navigate through the project, going from tickets to revisions to source code, and so on. The Timeline view is a powerful means of keeping tabs on all changes and updates. Any activity by users displays here, including activity involving tickets and changes to the source code repository (see Figure B.5).

The Roadmap view lets users track project progress through milestones (which can be synonymous with iterations, sprints or cycles). The Roadmap view gives a graphical view of the number of tickets closed compared to the total number of tickets for each

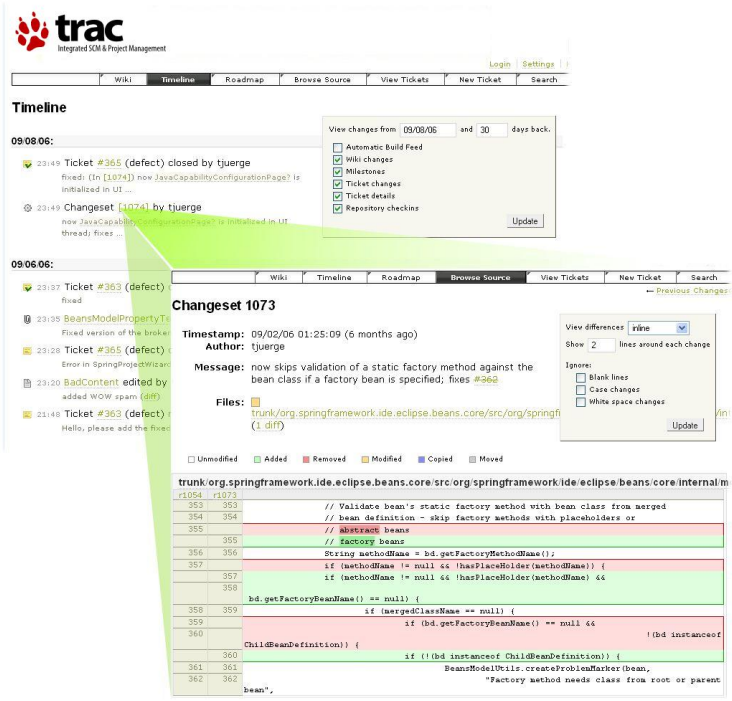

Figure B.5: The Trac timeline view.

milestone. Because tickets can represent tasks as well as bugs, this can also be a useful way to coordinate and track team activity and progress. More sophisticated reporting features are limited in Trac; there are no fancy graphs or charts, nor can users track ticket data over time.

B.2.6 Extensionsibility

An added bonus of Trac is that it is easily extensible via extension points provided within *trac.core*. The various extension points for the plugin functionality supported by trac are defined below:

trac.env.IEnvironmentSetupParticipant Allows plugins to participate in the creation and upgrade of the environment. Can be used to setup additional database tables or directories needed for the plugin to operate

trac.web.api.IRequestHandler Allows plugins to add handlers for HTTP requests.

trac.web.api.IRequestFilter Allows plugins to add filters to the HTTP requests.

trac.web.chrome.INavigationContributor Allows plugins to extend the navigation menus of the web interface.

trac.web.chrome.ITemplateProvider Extension point interface for components that provide their own ClearSilver templates and accompanying static resources.

trac.perm.IPermissionRequestor Plugins can use this extension point to define additional "actions" for the permission system.

trac.timeline.ITimelineEventProvider Allows plugins to contribute events to the timeline.

trac.mimeview.api.IHTMLPreviewRenderer Allows plugins to provide support for rendering specific content of a specific type as HTML (used for TracSyntaxColoring and image preview)

trac.wiki.api.IWikiChangeListener Allows plugins to observe creation, modification and deletion of wiki pages.

trac.wiki.api.IWikiMacroProvider Allows plugins to contribute WikiMacros to Trac.

trac.wiki.api.IWikiSyntaxProvider Plugins can extend this extension point to add custom syntax rules to the wiki formatting system. In particular, this allows registration of additional TracLinks types.

trac.ticket.api.ITicketChangeListener Extension point interface for components that require notification when tickets are created, modified, or deleted.

B.2.7 Summary

Trac is a solid lightweight issue-management solution well adapted to small teams, especially when Subversion is used. Trac is fully customisable. Because it's a wiki, users can modify the content of every page providing a possibility for knowledge management. Users can also tailor the look and feel by customizing the main ticket fields such as priorities, severities and ticket types and adding corporate logos. This customisability added with the plugin support - makes Trac a very functional and flexible solution.

B.3 JIRA

JIRA is a proprietary enterprise software product, developed by Atlassian, commonly used for bug tracking, issue tracking, and project management. JIRA is a widely used and well-regarded commercial issue-management tool used to manage bug tracking for many large scale open source and public projects. JIRA allows users to prioritise, assign, track, report and audit their 'issues': providing support for - software bugs, help-desk tickets, project tasks or change requests.

B.3.1 Synopsis

Like Bugzilla, JIRA is well adapted to large projects. JIRA handles multiple projects and project categories with ease, and allows users to set up permissions and various levels of security to limit who has access to particular projects, or even particular issues. JIRA is highly configurable - from directly within the administration screens, users can modify everything from issue types and priorities to the look and feel of the Web site.

B.3.2 Search Functionality

The JIRA search functionality (see Figure B.6) lets users perform full-text searches with filtering on key fields such as issue type, status, affected or fix versions, reporter, assignee, and priorities, as well as by date. The JIRA search is simpler and less cluttered than the equivalent Bugzilla Advanced Search screen. Search results are displayed in tabular form, with many visual cues.

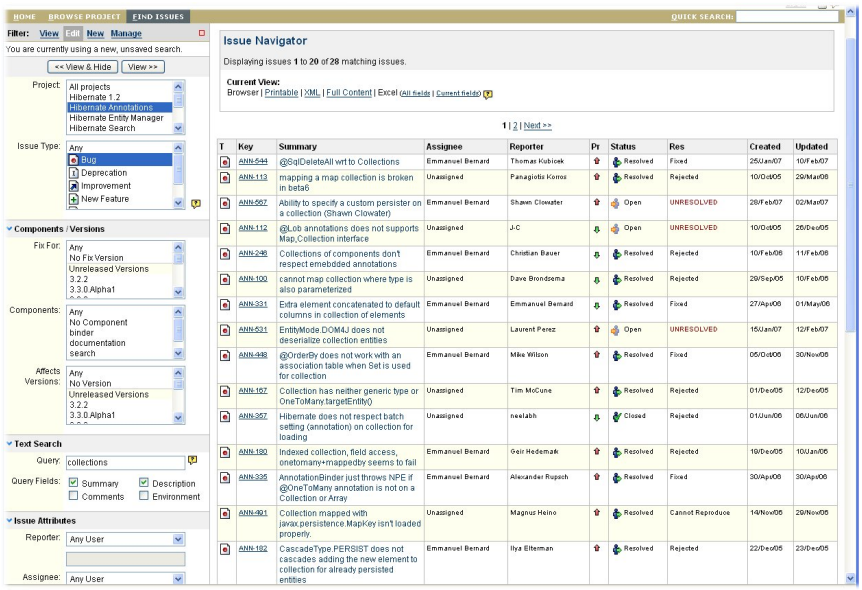

Figure B.6: Searching for issues in JIRA.

B.3.3 Change management

Creating new issues in JIRA is easy and requires minimum fuss. The workflow resembles that used for Bugzilla issues, though with a few minor differences. An “In Progress” bug, for example, is a bug that has been assigned to someone and is being actively worked on. Users can also tailor the workflow to their specific needs, by adding or removing steps in the workflow.

B.3.4 Workflow support

JIRA makes it easy to follow the entire life of a bug, issue, defect or feature request. The workflow engine lets users customise the path a bug takes. Using JIRA's customisable workflow engine, users can define individual workflows for departments, projects and even task types. Each workflow can have as many (or as few) steps as required.

B.3.5 Reporting

Reporting in JIRA is limited to search results: there are no graphs or bar charts, and there is no easy way of keeping track of time-related data such as the number of resolved issues over time. However, the JIRA user interface is a pleasure to use, and is intuitive. The home screen contains numerous graphical reports designed to give users a quick overview of the project's current status: the list of issues assigned to them, the "in-progress" issues, the number of open issues grouped by priority, and so on (see Figure B.7). A rich set of predefined issue filters and reports also eases the discovery of information.

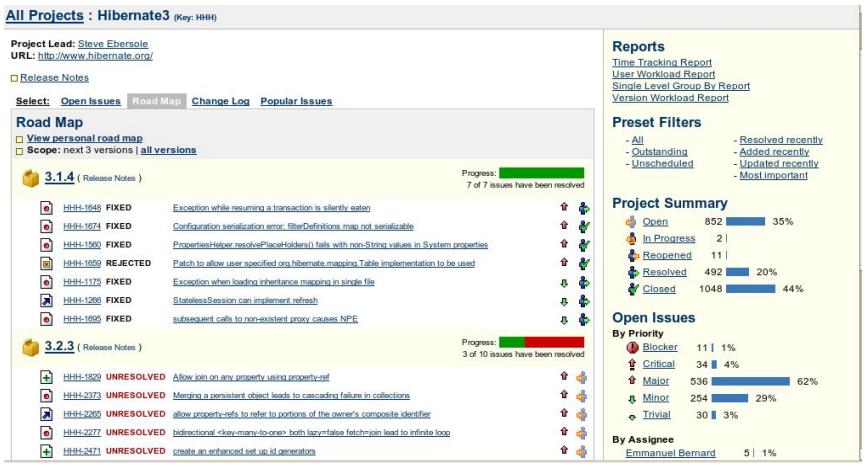

Figure B.7: The JIRA dashboard.

B.3.6 Summary

Jira is more than just an issue tracker, it is an extensible platform that users can customise to match to their business processes. JIRA lets users manage versions and product releases in a simple, intuitive manner. Users can define versions, track version releases and release dates, and generate release notes for a particular version with the list of all fixed and unresolved bugs for that release. However, as mentioned, JIRA, unlike Bugzilla and Trac, is a commercial tool. JIRA comes in three versions: standard, professional and enterprise, with some of the more advanced features such as project categories, configurable workflow and issue-level security reserved for the higher-level products. Prices range from $1,200 to $4,800 for a server license.

Appendix C

Traczilla Database Schema

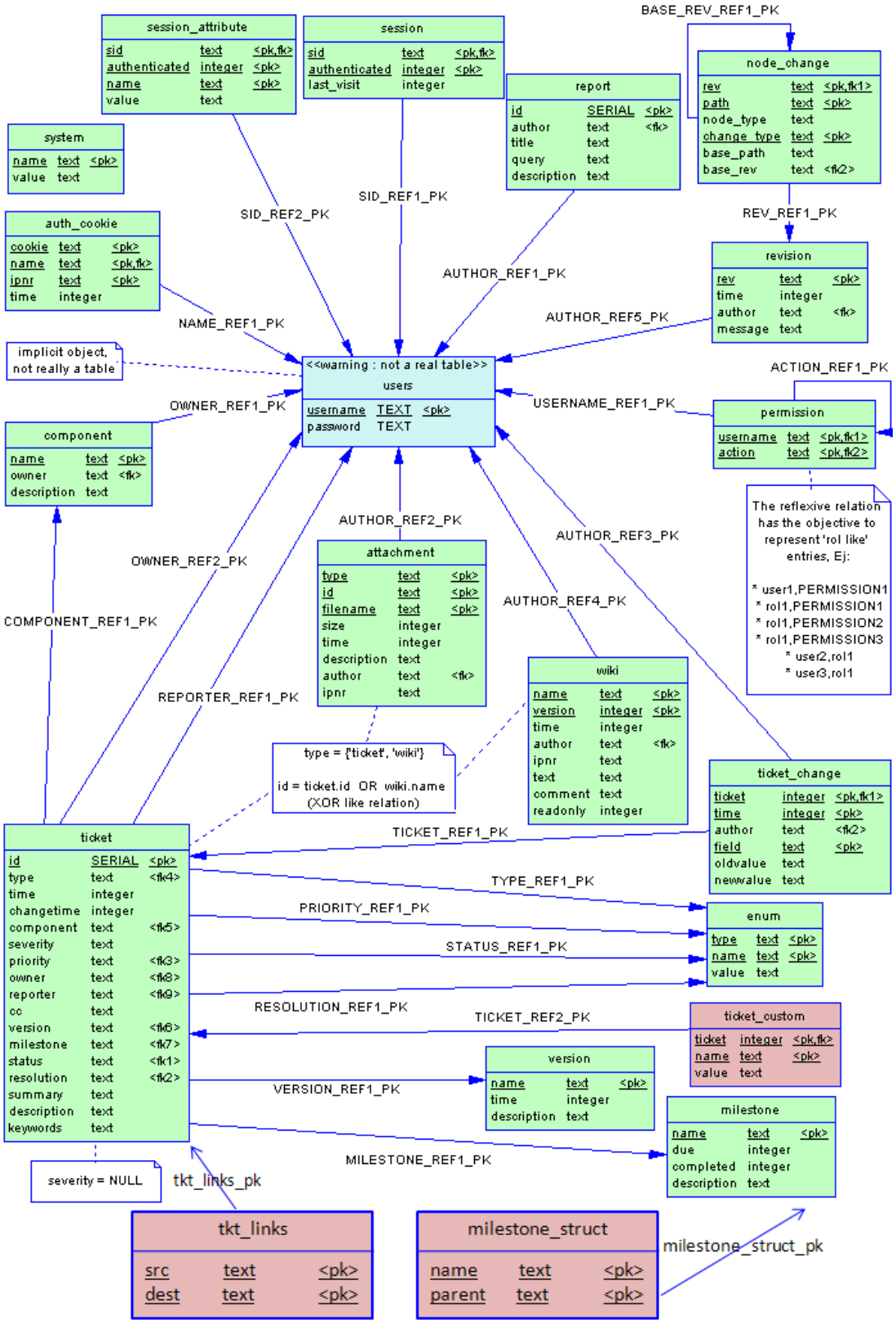

Figure C.1: Traczilla Database Schema, adapted from [27].

Bibliography

[1] IEEE Std 828-1990. Ieee standard for software configuration management plans. *Spring Software Engineering Standards Collection*, April 5 1991. 245 E. 47th St., New York, NY.

[2] Dave Abrahams. Trac Component Architecture, July 2007. http://trac.edgewall.org/wiki/TracDev/ComponentArchitecture.

[3] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and Limitations of Current Workflow Management Systems. *IEEE Expert*, 12(5):105–111, 1997.

[4] R.M. Baecker. *Readings in human-computer interaction: Toward the year 2000*. Morgan Kaufmann, 1995.

[5] Matthew P. Barnson. Anatomy of a bug, October 2006. http://www.bugzilla.org/docs/2.18/html/lifecycle.html.

[6] E.H. Bersoff, V.D. Henderson, and S.G. Siegel. Software configuration management. *ACM SIGSOFT Software Engineering Notes*, 3(5):9–17, 1978.

[7] G.A. Bolcer and R.N. Taylor. Advanced workflow management technologies. *Software Process: improvement and practice*, 4(3):125–171, 1998.

[8] J. R. Callahan, R. R. Khatsuriya, and R. Hefner. Web-based issue tracking for large software projects. *IEEE INTERNET COMPUTING*, 2(5):25–33, Sept.–Oct. 1998.

[9] T.H. Davenport. *Process innovation: reengineering work through information technology*. Harvard Business School Press, 1993.

[10] U. Dayal, M. Hsu, and R. Ladin. Business process coordination: State of the art, trends, and open issues. In *Proceedings of the 27th Very Large Databases Conference (VLDB 2001)*, 2001.

[11] N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. *interactions*, 8(5):30–38, 2001.

[12] J. Estublier. Software configuration management: a roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 279–289. ACM New York, NY, USA, 2000.

[13] M. Fowler and J. Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

[14] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153, 1995.

[15] R.B. Grady. Software Failure Analysis for High-Return Process Improvement Decisions. *HEWLETT PACKARD JOURNAL*, 47:15–24, 1996.

[16] V. Gruhn and J. Urbainczyk. Software process modeling and enactment: an experience reportrelated to problem tracking in an industrial project. In *Software Engineering, 1998. Proceedings of the 1998 International Conference on*, pages 13–21, 1998.

[17] C. Hagen and G. Alonso. Exception handling in workflow management systems. *IEEE Transactions on software engineering*, 26(10):943–958, 2000.

[18] Brian Hermann and Jim Marshall. Are you ready to deliver? to ship? to test? *Software Technology Support Center, The Defense Journal of Software Engineering*, 1998.

[19] J.D. Hole. Email overload in academia. *Unknown*, 2008.

[20] D. Hollingsworth et al. *The workflow reference model*. Workflow Management Coalition, 1995.

[21] M. Jansen-Vullers and M. Netjes. Business process simulation–a tool survey. In *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, October*, 2006.

[22] GE Kaiser, IZ Ben-Shaul, SS Popovich, and SE Dossick. A metalinguistic approach to process enactment extensibility. In *Software Process, 1996. Proceedings., Fourth International Conference on the*, pages 90–101, 1996.

[23] P.J. Kammer, G.A. Bolcer, R.N. Taylor, A.S. Hitomi, and M. Bergman. Techniques for Supporting Dynamic and Adaptive Workflow. *Computer Supported Cooperative Work (CSCW)*, 9(3):269–292, 2000.

[24] J. Keyes. *Software configuration management*. Auerbach Publications, 2004.

[25] A. Leon. *A Guide to software configuration management*. Artech House, Inc. Norwood, MA, USA, 2000.

[26] A. Leon. *Software configuration management handbook*. Artech House, Inc. Norwood, MA, USA, 2004.

[27] Johans Marvin and Taboada Villca. Overview of Trac 0.10 and 0.11 Database Schema, July 2007. http://trac.edgewall.org/wiki/TracDev/DatabaseSchema.

[28] S. McCready. There is more than one kind of workflow software. *Computerworld*, 2:86–90, 1992.

[29] C. Mohan. Recent Trends in Workflow Management Products, Standards and Research. *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, 164:396–409, 1998.

[30] R.S. Pressman and D. Ince. *Software engineering: a practitioner's approach*. McGraw-Hill New York, 1982.

[31] M. Reichert and P. Dadam. ADEPT flexsupporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[32] H.A. Reijers, JHM Rigter, and W.M.P. van der Aalst. The case handling case. *International Journal of Cooperative Information Systems*, 12(3):365–392, 2003.

[33] K.D. Swenson and K. Irwin. Workflow technology: trade-offs for business process reengineering. In *Proceedings of conference on Organizational computing systems*, pages 22–29. ACM New York, NY, USA, 1995.

[34] Walter Tichy, editor. *Configuration Management (Trends in software)*, volume ISBN 0-471-94245-6. John Wiley, 1994.

[35] W.M.P. van der Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.

[36] WMP van der Aalst, HT de Beer, and BF van Dongen. Process mining and verification of properties: An approach based on temporal logic. *Lecture notes in computer science*, 3760:130, 2005.

[37] WMP van der Aalst and S. Jablonski. Dealing with workflow change: identification of issues and solutions. *COMPUTER SYSTEMS SCIENCE AND ENGINEERING*, 15(5):267–276, 2000.

[38] W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: A new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.

[39] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag Berlin Heidelberg, 2007.

[40] D. Whitgift. *Methods and tools for software configuration management*. John Wiley & Sons, Inc. New York, NY, USA, 1991.

[41] S. Whittaker and C. Sidner. Email overload: exploring personal information management of email. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, pages 276–283, 1996.

[42] M. Zur Muehlen and D.T.Y. Ho. Risk Management in the BPM Lifecycle. In *BPM*, pages 454–466. Springer, 2005.