

Juha Vitikka

Supporting Database Interface Development with Application Lifecycle Management Solution

VTT PUBLICATIONS 714

Supporting Database Interface Development with Application Lifecycle Management Solution

Juha Vitikka



ISBN 978-951-38-7353-0 (URL: <http://www.vtt.fi/publications/index.jsp>)

ISSN 1455-0849 (URL: <http://www.vtt.fi/publications/index.jsp>)

Copyright © VTT 2009

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 5, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 5, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax + 358 20 722 4374

Technical editing Leena Ukaskoski

Juha Vitikka. Supporting Database Interface Development with Application Lifecycle Management Solution [Tietokantarajapinnan kehittäminen ohjelmiston elinkaaren hallinnan avulla]. Espoo 2009. VTT Publications 714. 54 p.

Keywords ALM, SCRUM, TFS, data transport formats

Abstract

Controlling a software project has a major effect on the project's productivity, expenses and the quality of a project's product and code. This work investigates Application Lifecycle Management which considers how software, the software process and its different phases are controlled. With the help of Application Lifecycle Management a working database interface for embedded testing framework has been developed.

In this Microsoft Team Foundation Server is used for managing the software project. As a software process SCRUM is used by utilizing SCRUM for Team System process template developed by Conchango. The process template is customized to bring support for requirement management in to it. The customized process template is used in the demo project, in which a database interface for embedded testing framework is developed. Thus the process template customization is tested in practice and experiences of ALM, Microsoft TFS and SCRUM process are gathered.

During development of the embedded testing framework, which is one sort of generic data gathering tool, one must pay attention to many issues such as data transport methods and formats, database solutions, data export methods and integration. Database interface software, called Probe DB, is developed according to the requirements of the customer. Own CSV and binary data transport formats and XML format for export functionality are designed, and interfaces for file and TCP/IP import and for Eclipse IDE are developed. Software is coded with Python and MySQL will serve as database solution.

Tiivistelmä

Ohjelmistoprojektin hallinta vaikuttaa merkittävästi projektin tuotteen ja koodin laatuun, tuottavuuteen ja kuluihin. Työssä perehdyttiin ohjelmiston elinkaaren hallintaan (ALM), joka käsittelee sitä, miten ohjelmistoa, ohjelmistoprosessia ja sen eri vaiheita hallitaan. Ohjelmiston elinkaaren hallintaa apuna käyttäen kehitettiin toimiva tietokantarajapinta sulautettuun testauskehikkoon.

Tässä työssä ohjelmistoprojektin hallintaan käytettiin Microsoftin ohjelmiston elinkaarenhallintatyökalua Team Foundation Serveriä. Ohjelmistoprosessina käytettiin SCRUMia Conchangan TFS:ään kehittämän SCRUM for Team System -prosessipohjan avulla. Prosessipohjaa muokattiin vaatimustenhallinnan mukaan tuomiseksi. Muokattua prosessimallia sovellettiin projektissa, jossa kehitettiin tietokantarajapinta sulautettua testauskehikkoa varten. Näin testattiin prosessimallin muokkauksen soveltuvuus ja samalla kerättiin kokemuksia ohjelmiston elinkaaren hallinnasta, MS TFS:stä ja SCRUM-prosessista.

Sulautettu testauskehikko voidaan katsoa geneeriseksi tiedonkeruutyökaluksi, jollaiseen tietokantarajapintaa kehittäessä pitää ottaa huomioon useita seikkoja, kuten tiedonsiirtomenetelmät ja formaatit, tietokantaratkaisu, tiedon vieminen sekä integrointi. Tietokantarajapintasovellus, nimeltään Probe DB, kehitettiin asiakkaan tarpeiden pohjalta. Sovellukselle suunniteltiin omat CSV- ja binääri-muotoiset tiedonsiirtoformaatit sekä XML-formaatti ulosvietävän tiedon määrittämiseen ja kehitettiin tiedosto- ja TCP/IP-rajapinnat testauskehikkoon sekä TCP/IP-rajapinta Eclipse IDEen. Sovellus koodattiin Pythonilla ja tietokantaratkaisuksi valittiin MySQL. Kehitystyön tuloksena syntyi toimiva tietokantarajapinta osaksi sulautettua testaustyökalua, Probe Frameworkiä.

Preface

This work was done in VTT as a contribution to the TWINS project.

I would like to thank my master thesis work instructor at VTT, Jukka Kääriäinen, and my supervisor at Oulu University Professor Juha Röning for professional work instruction, and my colleges Teemu Kanstren and Markku Pollari for providing valuable help during my work. Thanks also to other colleges, friends and family who have supported me along the way.

Oulu, April 30, 2008

Juha Vitikka

Contents

Abstract	3
Tiivistelmä	4
Preface	5
List of symbols	8
1. Introduction	9
2. Application Lifecycle Management	11
2.1 Lifecycle Models	11
2.1.1 Traditional Models and Agile Models	12
2.1.2 SCRUM	14
2.2 ALM Concepts	16
2.3 ALM Tools.....	18
2.4 MS TFS.....	18
2.4.1 TFS from ALM Framework Perspective	19
3. Data Transport Formats and Databases of Data Gathering Tools	22
3.1 Data Transport.....	23
3.1.1 Data Import.....	23
3.1.2 Export.....	24
3.2 Databases.....	24
4. Work Assignment.....	26
4.1 Probe Database Development Needs	26
4.1.1 Functional Requirements	28
4.1.2 Technical Requirements.....	28
4.2 MS TFS Research Needs.....	29
5. Implementation	31
5.1 Development Environment Setup	31
5.1.1 TFS Installation And Configuration.....	31
5.1.2 Process Template Installation And Customizations	32
5.1.3 Project Setup.....	36

5.1.4	Visual Studio and Python Integration	37
5.2	Probe Database Development.....	37
5.2.1	Testing.....	44
5.2.2	Limitations	44
5.3	Software Development with TFS	45
5.3.1	Working with MS TFS.....	45
5.3.2	Software Configuration Management.....	46
5.3.3	SCRUM	47
6.	Discussion.....	49
6.1	TFS Usage Experiences	49
6.2	Experiences about Database Interface Development	50
6.3	Probe Database Future Development	51
7.	Conclusion	52
	References.....	53

List of symbols

ALM Application Lifecycle Management

CM Configuration Management

CMM Capability Maturity Model

DB Database

HW/SW Hardware/Software

RM Requirement Management

RUP Rational Unified Process

TFS Team Foundation Server

TCP/IP Transmission Control Protocol / Internet Protocol

VTT Technical Research Centre of Finland

XP Extreme Programming. A software engineering methodology or a software process.

1. Introduction

A successful software project requires the management of many steps of the software process and a large amount of artefacts such as requirements, code, documents and tests. Software development is done in logical steps, including definition, design, development, testing, deployment and management. Controlling these steps, developed artefacts and software project in overall is called Application Lifecycle Management. It also facilitates taking care of communication between team members and work coordination. Nowadays software projects are also expected to produce better quality products faster and cheaper. Application Lifecycle Management is claimed to improve code quality and team productivity, thus accelerating development and reducing maintenance time. It is also required in the development of some mission or safety critical systems.

The purpose of this work is to develop a database interface for an embedded testing framework with the help of ALM and ALM tool. Simultaneously the use of ALM in a software project will be investigated. An ALM tool, Microsoft TFS, will be tested for software project management by carrying out a demo project during which a database interface will be developed. There is also a need for some customizations to TFS which will be tested in practice during the demo project. A database interface is needed for embedded testing solution to enable importing data from embedded tests to database, and for getting test information out of the database in the desired format.

There are many commercial ALM tools. ALM tools vary from separate tools to integrated and central database solutions. ALM can be applied to software development using different kinds of lifecycle models. A theoretical part of this work investigates ALM in general, ALM tools and different lifecycle models.

This work is done as a part of the ITEA-TWINS project. The TWINS project investigates co-design problems of integrated hardware and software development. The main problems in this development mode are co-specification and the

1. Introduction

allocation of requirements, co-optimization of HW/SW architectures, lifecycle management and configuration management of evolving products and components (hardware, software), and the improvement of testing of multidisciplinary products. Results expected from the project include solutions and inventories to support HW/SW co-design based on industrial needs and experiences.

2. Application Lifecycle Management

Application lifecycle management is a rather new concept so there are not many publications that deal with the term. Its origins are in the history of source control management and integrated development environments. There are many definitions for ALM and maybe for that reason companies do not always know what to expect from ALM. According to one definition, ALM means the coordination of development life-cycle activities, including requirements, modelling, development, build, and testing [1]. That involves supporting the processes that span these activities, managing relationships between development artefacts used or produced by these activities and reporting on progress of the development effort as a whole. ALM handles artefacts of lifetime activities and keeps them synchronized for traceability and reporting purposes, for instance. Shaw defines that ALM is neither a product nor a process – it is a way of doing business within application development [2]. In this thesis, ALM is considered from the development lifecycle point of view, as described by Schwaber [1]. ALM, although it is a rather new field, is used more and more in industry. Especially in the aerospace industry, where they develop a lot of mission and safety critical software, it has become an important tool for software development [3, 4].

2.1 Lifecycle Models

Software lifecycle models describe how to manage different activities of software development process. These are also called software processes or software development processes. Though there is a great variety of different kinds of software lifecycle models most of them have similar patterns and work phases. Usually they include at least these four work phases: gathering requirements, designing, implementing and testing.

2.1.1 Traditional Models and Agile Models

There is a great variety of different kinds of software lifecycle models based on different structures and principles. Every model has its advantages and disadvantages and selecting a model should be done depending on what kind of project is going to be carried out. There is no clear agreement on how to distinguish agile models from traditional models [5]. Traditional models are said to be more discipline and plan-driven, whereas agile models are more unplanned and undisciplined, emphasizing agile principles.

Some well known traditional models are the waterfall model, spiral model, V-model, RUP and CMM. The waterfall model shown in Figure 1 is a well known classic lifecycle model with a very simple structure. In waterfall processes, software development flows downwards through development phases. After each phase there is an evaluation of the project's progress to determine whether it is worthwhile continuing, or should be stopped. [6, 7]

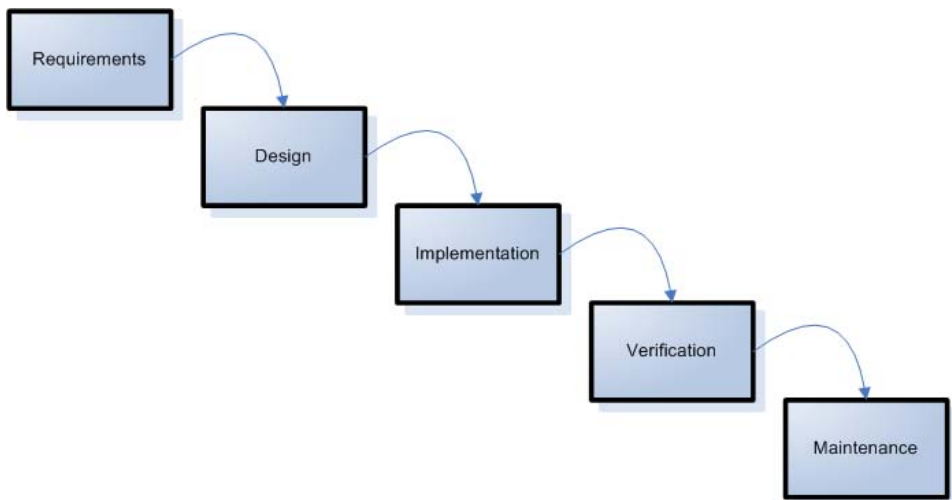


Figure 1. Waterfall Model.

In iterative software processes, software is developed in iterations. During iteration a proportion of the software is developed so that the software gradually evolves into the final product. Iteration consists of work phases defined by process model. In traditional models iterations can be quite long, even months, whereas in agile models iterations are usually a few weeks long. We get a simple

example of an iterative model by repeating the stages of the waterfall model in iterations. Figure 2 shows another iterative model, called the spiral model.

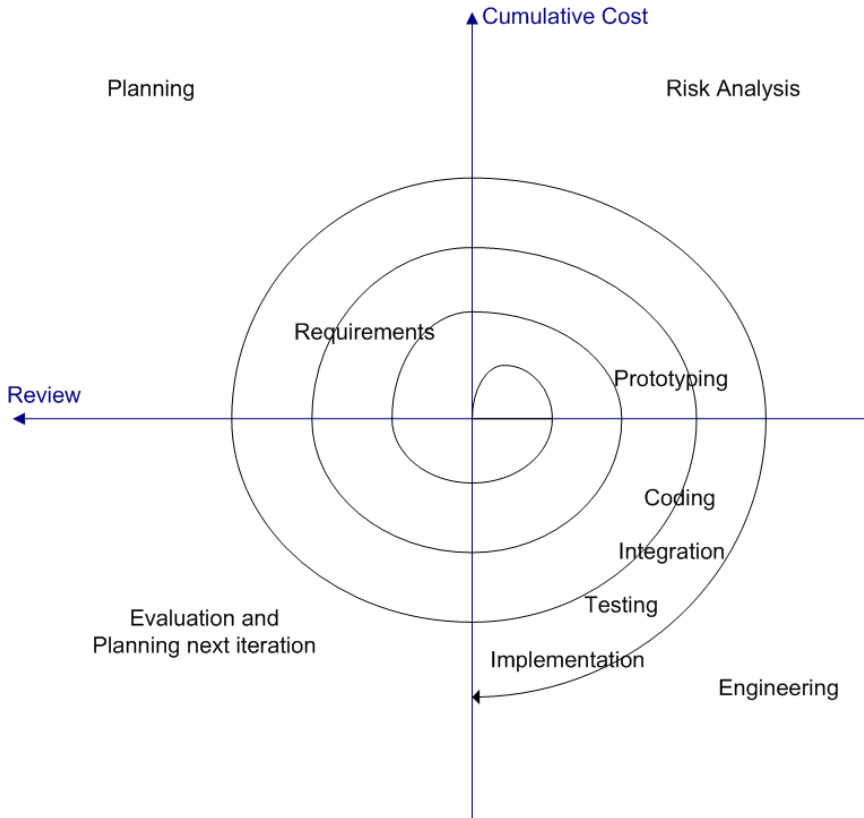


Figure 2. Spiral Model.

The spiral model starts with planning and gathering requirements, after which risk analysis is done. This involves making the design and prototype of the product. If continuing the project seems reasonable according to risk analysis and prototype, the process moves to engineering phase. After the product version is ready it is evaluated and the next iteration planning starts.

Agile software development models emphasize iterative, incremental, light weight and agile approach to software development. Most agile models aim at reducing risk by shortening the development time. They are based on agile principles defined in the agile manifesto created in 2001 by 17 agile experts. Some of the main principles of agile methods are rapid and frequent delivery of useful

software, face-to-face communication, simplicity, self-organizing teams, daily cooperation between business people and developers, regular adaptation to changing circumstances. The working software is considered as a measure of progress. Most models also favor small teams with people located close to each other and little documentation. For example, SCRUM and XP are software development models used for agile development. [5]

Agile methods have their advantages and disadvantages and they are better suited for particular types of projects. Benefits from agile methods are, for example, adaptability, better code quality and reduced risks. However, fast requirement gathering and rapid changes during the project may result in a quite different product than the optimal solution. Suitability of agile methods can be examined from many different perspectives. Agile methods are more adaptive to changes, so they are more suitable for products that are subject to rapid and frequent changes. This, however, costs reliability so traditional methods might be better suited for products that have criticality, reliability and safety requirements. Agile methods work better in projects with small teams, because in larger teams face-to-face communication becomes more difficult.

2.1.2 SCRUM

SCRUM is an iterative and incremental software development process. It defines a set of roles and practices to carry out a software development process which produces a working product increment with every iteration. SCRUM's roles are ScrumMaster, Product Owner, Stakeholders, Team Members and Users. Iterations, also called as sprints, are usually 15–30 days, focusing on high quality code and product's customer value. In addition to daily development work SCRUM iteration has other activities: Sprint Planning Meeting, Sprint Review, Sprint Retrospective and Product Backlog Update. Every sprint also aims to deliver a working product increment that brings value for the customer. Before the start of the SCRUM guided process there is a set of activities for preparing project. [5, 8]

The SCRUM process is guided by the ScrumMaster who takes responsibility that the process is progressing correctly. The ScrumMaster's tells other team members their roles in the project and how they should work in their roles. He supervises that SCRUM rules are followed and tries to remove all impediments compromising the efficiency of SCRUM. The ScrumMaster's responsibility is to improve practices, enable close cooperation, shield the team from external inter-

ferences and thus maximize the team’s productivity. He also teaches the product owner how to best benefit from SCRUM and maximize profit. [8]

The product owner is usually an employee from the customer’s company. His role is to represent everyone’s interest in the project, and to ensure that the project makes a profit. He should have a good picture of the product and its features and requirements. The product owner must understand the business value of features and requirements of the product being developed to make sure that it is going to be profitable. His tasks include defining product features, prioritizing them and gathering input from customers and stakeholders. He also decides on the product release date and accepts or rejects work results. [8]

Team members in SCRUM are a group of cross-functional people with skills to develop a product according to requirements. Usually a team, consisting of about seven people, includes an analyst, designer, a quality assurance person, coder and documentation person. They specify iteration goal and try to reach it following SCRUM rules. The ScrumMaster guides them to follow SCRUM work patterns but they organize their work themselves. [8]

Before SCRUM iterations start there is a preparation phase also called Sprint 0. This preparation phase involves making a business case, getting funding, building a vision of product, assembling a team and creating initial product backlog and a release plan. The business case is important for figuring out if project is going to be profitable and depending on that, whether it is worthwhile to carry it out at all. The vision should be a short description of the product or project’s goal that should be clear for all team members. The initial product backlog is created before the first sprint, gathering requirements from the product owner,

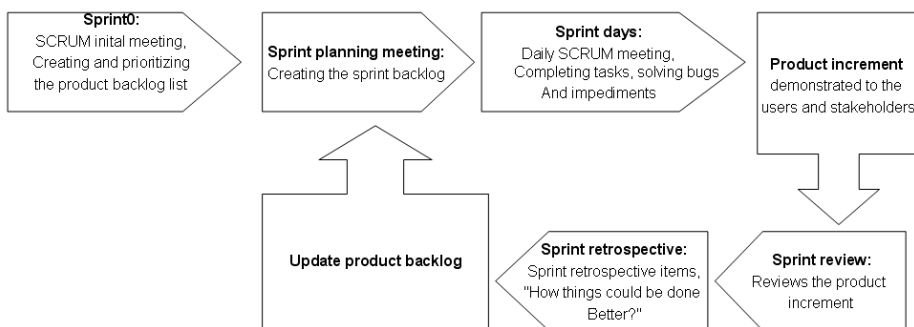


Figure 3. SCRUM Process.

business case or elsewhere. After team roles are identified, a kick-off meeting is held where the backlog is reviewed and technical issues, team members' duties and scope of project are discussed. [8]

SCRUM process, shown in Figure 3, forms of SCRUM activities which are repeated in every sprint. Before starting the SCRUM sprint, the product backlog must be updated and prioritized with the product owner. The updated product backlog is presented in the sprint planning meeting where the team selects with the product owner the product backlog items to build in the next sprint, according to product backlog prioritization. The second part of the sprint planning meeting is sprint workload planning. The team designs how to implement the functionality that they have selected and the architecture required. Work is then split into tasks, and tasks are given estimates of the required work time needed to complete them. The team is expected to get all work done that they have selected during the sprint, so that estimates should not dramatically exceed team capacity. With complete sprint backlog the daily SCRUM work can start. Every day is held a short daily SCRUM meeting during which team members tell what they have done and are going to do next and report impediments to ScrumMaster for removal. At the end of sprint team should deliver a product increment which is demonstrated to the users and stakeholders. It should be working software with the new functionality selected to be implemented during the sprint. At last a sprint review is held where team represents management, customers, users and product owner the sprint goal, the product backlog committed to, the product backlog completed and how the sprint has overall gone. The team tells tales of their working during the sprint, what went right or wrong and to what direction is product being developed. With this information the management, users, customers and product owner can make decisions what to do next. After the sprint a sprint retrospective is held with team members and ScrumMaster. At sprint retrospective recent sprint is discussed and improvement possibilities are determined. Team considers its work environment, work procedures, activities and everything that is affecting having a successful sprint. Issues of what went well and what did not are gathered and used to improve next sprints. [8]

2.2 ALM Concepts

ALM framework introduced by Kääriäinen and Välimäki describes the principal elements of ALM. The elements can be seen in Figure 4: Creation and manage-

ment of lifecycle artefacts, traceability, communication, reporting, process automation and tool integration. [9]

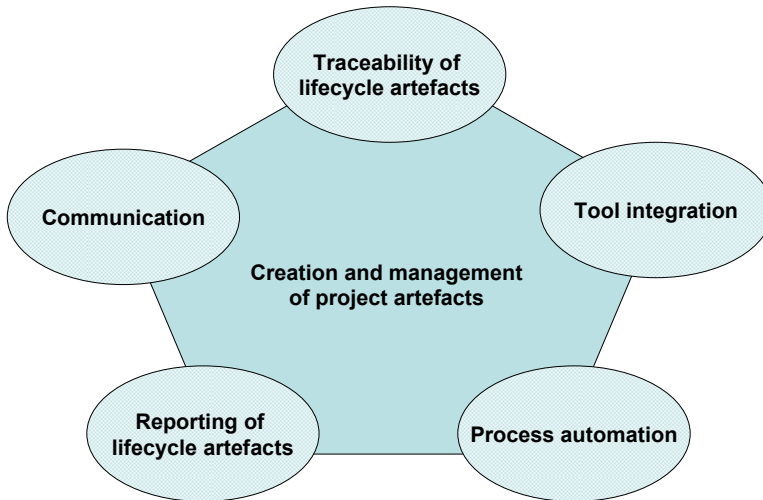


Figure 4. ALM Framework.

Creation and management of lifecycle artefacts means how project lifecycle artefacts, such as project's requirements, documents, source code, builds, project control items, etc., are created and managed in ALM solution. Managing these artefacts includes creating, identifying, storing and versioning them in different project phases. [9]

Traceability in ALM solutions means how traceability information between project lifecycle artefacts is gathered and maintained. Information about relationships between artefacts gives visibility to projects' lifecycle and facilitates many ALM operations like reporting and change impact analysis. [9]

Communication is very important element of ALM. Project members and other people involved need to exchange information efficiently and securely. This element describes how ALM solution handles communication and information sharing. [9]

The reporting element means how reporting is supported. Information about process and configuration items need to be gathered, processed and presented. [9]

Process automation means what kind of support and automation is offered for process coordination. There are many work tasks in processes that can be automated, thus saving work time for other activities. [9]

Tool integration discusses how different tools for ALM are integrated together, or what level of support there is for integration [9].

2.3 ALM Tools

There are many freeware and commercial ALM tools available. There are also some tools and frameworks which support and facilitate developing and integrating ALM tools. Such are, for example, Eclipse Tool Integration Framework and ALF, Application Lifecycle Framework [10]. ALM solutions can be divided in three types: Single vendor platform, Multi-vendor platform and single repository solution. A single vendor platform is an interoperability framework defined by single vendor. ALM practitioners and other vendors can build integrations to that platform, but usually have no means to influence the actual platform. Commercial ALM solutions, “Borland’s ALM solutions” and “IBM Application Lifecycle Management solutions”, are this kind of ALM platforms. A multi-vendor platform is developed by many vendors in an open-source community. “Compuware application delivery management and IT portfolio management solutions” is an example of a multi-vendor platform. The third type of ALM solution, Single repository, is an ALM solution built on a single repository, containing a complete set of ALM tools. One repository facilitates traceability and reporting. Microsoft Team Foundation Server is based on single repository solution, although it also can be considered as a single vendor platform. [2]

2.4 MS TFS

Microsoft Team Foundation Server is Microsoft’s ALM solution. It’s an integrated ALM suite covering all the main aspects of ALM. It’s available as a stand-alone version and as a server side platform. As a server platform it supports different server topologies from a single server platform to very complex solutions. Thus TFS is highly scalable for different sizes of projects and companies. In this work MS TFS is used as an ALM tool for a software development project developing a database interface called Probe DB. TFS was chosen because it is used by the TWINS project partner who also has a need to investigate the use of TFS.

Architecturally TFS consists of three tiers: Application Tier, Data Tier and Client Tier. The data tier consists mainly of Microsoft SQL Server 2005 data-

bases and data stores. It can be installed on the same or separate server machine as TFS application tier. The data tier can not be accessed directly but the data is accessible through the application tier. The application tier runs TFS applications and web services and offers web services API through which client programs can integrate with TFS.

TFS Build and TFS Proxy are also included with TFS. TFS Build offers support for build automation and for sharing builds for testers. TFS proxy is used to cache source control files. This is useful in distributed software development when the actual TFS server is far from development site. Proxy is located near development site and it caches source control files making it faster to work with them.

2.4.1 TFS from ALM Framework Perspective

The ALM framework can be used to characterize how MS TFS handles the different aspects of ALM. This framework consists of six main areas of ALM: Creation and management of lifecycle artefacts, traceability, communication, reporting, process automation and tool integration [9].

The TFS solution for creation and management of project artefacts is based on integrated ALM suite and central database. All project artefacts can be created and controlled from Visual Studio using an integrated TFS client, Team Explorer. Project artefacts are stored by the TFS data layer, which uses MS SQL Server 2005 as information storage. Permission to project artefacts is handled through TFS group membership and security settings. Project artefacts in TFS include work items, builds, source code, reports and documents.

Work items are used for tracking work in the project and communicating project information between project members. The basic project management items such as requirements, tasks, bugs and issues, all appear as work items in TFS. Work items are process-template specific, so they may have different names and properties depending on process template used. Work items are not versioned but their change history is recorded. They are stored in SQL Server work item database. It is possible to create new work item types or modify the existing ones if necessary, but that may require some study.

TFS has a source control repository for source code and documents. TFS's source control system is based on change sets. Changeset is a logical container in which TFS stores everything related to a single check-in operation: file and folder revisions, links to related work items, check-in notes, a comment, policy

2. Application Lifecycle Management

compliance and system metadata such as owner name and date/time of check in. All data related to version control is stored in the SQL server source repository database through TFS data layer.

TFS has a build server which builds the builds and keeps them shared for testers. TFS build server works as a centralized place for builds. A build in TFS can also appear as work item that can be linked to changesets.

Documents can also be shared in the document library of the TFS project portal. Document library keeps documents versioned if versioning is turned on.

Traceability of lifecycle artefacts in TFS is partly automated, but it's also possible for the user to create links manually. Links between various types of project artefacts can be created quite freely and without constraints. Traceability data between source control objects and tasks is gathered automatically, if this check-in policy is turned on. When a work item is created from another, links are also automatically created. Automated traceability depends on the process template and its working procedures. Mainly it is left for project members to work in such a way that traceability data is gathered correctly. These procedures are described in process templates process guidances.

Reporting of lifecycle artefacts in TFS is supported. There's a report portal for project reports accessible with a web browser. Process templates include many useful reports and they can also be customized for a project's needs. Microsoft provides tools for report customization [11].

Communication is one of the main aspects of TFS. In the TFS project the project portal works as the central point of communication. Work items, project alerts and reports also facilitate communication. To ensure secure communication security settings define who can access and modify which information.

Work items are used to share information about projects' requirements, tasks, issues, bugs and so on. Work items are linked together, and with other lifecycle artefacts and their change history is recorded. Reports gather data about project artefacts and provide useful information about project's progression.

Project portal works as a central point of communication in TFS. Project portal is accessed with a web browser and has many tools for information sharing. There project members can share documents, discuss issues on message boards and share information about announcements, events and links. The project portal is customizable and process templates have their own customizations to do it.

Project templates in TFS provide process support/automation. From a technical view point process templates define project management items and their properties, state transformations and relations. Process templates also include

process guidance, reports and their own customizations to project portal. Process guidance explains process models' working procedures, roles, work items and reports.

The TFS application tier provides web services through which client applications can integrate to TFS. There are many commercial and open source tools that integrate with TFS. Some Microsoft's products, e.g. Microsoft Project and Excel, are meant to be used with TFS as an essential part of project management. There's also integration with Windows SharePoint Services and SQL Server. Many open source tools integrating with TFS can be found from Microsoft's open source community CodePlex.

TFS integrates with some windows server's features like active directory, workgroup and SMTP. SMTP integration enables sending email notifications from TFS and Active Directory and workgroup integration provides quite useful help for controlling users and groups in TFS.

3. Data Transport Formats and Databases of Data Gathering Tools

Probe Framework can be considered as a generic data gathering tool in which the data transport techniques and database solutions play an essential role. Data import and export functionalities need their own data transport methods and formats, so different solutions are considered here. The database solution has an effect on the actual database interface programming, but it does not affect much architecture or other parts of the software. Figure 5 gives an overview of Probe DB's architecture.

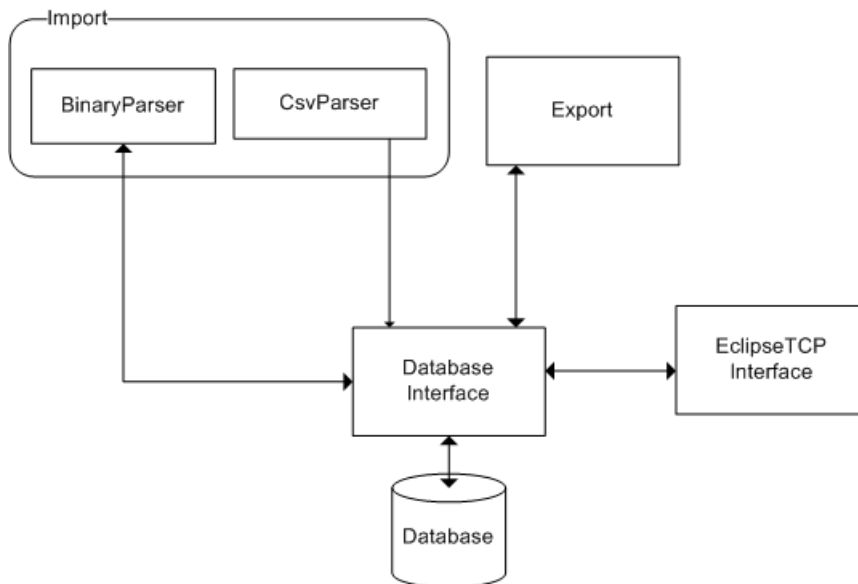


Figure 5. Overview of Probe DB's architecture.

3.1 Data Transport

Efficient data transport techniques are important in data gathering tools like Probe DB. Data is gathered in an embedded device and it must be imported to the database, so there is a need for import methods. Various data transfer techniques can be used, for example transfer using TCP/IP connection or file transfer [12]. The format used for data transfer also plays an important role. The gathered data is used in different third party software, test analysis software for example, so it must be possible to export data in different formats that these third party software understand.

3.1.1 Data Import

Getting the gathered data from the data gathering tool to the database requires a data transfer technique and a data format. Transfer can be done, for example, through TCP/IP connection or using a file. For transfer format there are several possibilities that can be used. One can design a specific format that is tuned to meet the needs of the software, or use some of the existing standard formats like XML based GXL [12].

Data transfer methods in data gathering products are often either transport over a network or transport using a file. A transport file is created automatically or manually in the software from which the data is going to be exported. The file is then read automatically or manually to software into which data is imported [13]. Transporting data over a network connection is currently quite popular. A network connection is established and used to transfer the data. TCP/IP is usually the protocol used for this but there are other possible protocols, like UDP, too.

Gathering data in a device undergoing testing should be as efficient as possible so that testing itself would not cause a probe effect, meaning to alter the behaviour of the system and affecting the test results [14]. This is especially significant factor in systems with real requirements, and in embedded systems with limited resources. Thus the data format should be designed in a way which does not require too many resources when used. There are a few existing standard formats that could be used for data transfer but they are not optimal in efficiency and resource usage. Some format standards, like GXL for example, are based on XML, which is quite slow compared to fixed length data field formats. If one designs his own data format for software with the intention to tune it for optimal performance, it is better to make it a delimited format. These kinds of formats

3. Data Transport Formats and Databases of Data Gathering Tools

are usually based on CSV or binary format. CSV, Comma Separated Values, is a simple way of formatting data. In CSV format values are in their fixed places separated with a separator character which is usually ‘;’. To design as optimal and compact a format as possible it is better to use binary format. Binary format codes all data values in binary. In CSV format values are in text format, for example number 10 appears in the CSV file as characters 1 and 0 consuming two bytes. In binary format 10 would appear as hexadecimal value 0x0A requiring only one byte. Thus binary saves space and is usually much faster to read. One example of need for efficient data transport is SEAT (Software Exploration and Analysis Tool) which uses a metamodel called CTF.

3.1.2 Export

When data may be used by different third party software there is a need for various export formats or the possibility to define ones own export format. Different software products understand data in different formats, and they may require manipulation, scaling and filtering data depending on what kind of data it is, and how it is analyzed. An example of analyze software used to analyze test data from the Probe Framework is Daikon Tool [15]. It is a tool for detecting likely invariants and it can read data from CSV files and Excel spreadsheets among other ways of getting data.

3.2 Databases

The database is an essential part of Probe DB, used to store data coming from the Probe Network. Many possible choices for database solution are available, ranging from relation databases to object oriented databases. Different database solutions are good for different things. Relation databases are popular and commonly used for many purposes. They store information to database tables which have relational links between them. Object oriented databases use objects to represent information. There are also post-relational databases which have tables and relations but do not have the information principle, requiring that all information is represented by data values in relations, as a constraint. MySQL is an example of relational database and Zope is an example of an object oriented database [16, 17].

Relational databases are databases that work according to a relational model storing their data in tables. Information in tables is organized to columns and

rows, and individual rows are often identified using a key column. Between tables there can be links which relate rows of information in one table to rows of another table. These links are implemented using some columns as foreign keys which reference foreign key columns in other tables. In a relational database, information is well organized and retrieving and combining information is relatively easy. Usually SQL, Simple Query Language, is used for retrieving and manipulating information in relational databases. Thus, SQL being a widely used industrial standard, relational databases are quite well supported by other software like reporting, OLAP and backup software, and they are also easily interfaced with many programming languages. However, the relational model encounters some difficulties with object-oriented programming languages due to differences in data representation, encapsulation and data types. [18, 19]

Object databases represent their information as objects the same way as in object-oriented programming languages. This facilitates developing database applications with an object-oriented programming language, because object database objects can be used as normal programming language objects. Object oriented databases are navigationally based, meaning that the information is found by following references through objects. For some data specific kind of searches this works quite fast, but general-purpose queries are often slower and more difficult to formulate when compared to relational databases. Object oriented databases are not that well supported by other software. [19]

In this work MySQL was chosen for the database solution, first of all because it is open source and free. Object based database Zope was also considered as one or possibly even a parallel alternative but the developer and customers were both more familiar with relational databases. Also the wider support for MySQL database SQL in other software possibly used for reporting purposes was one key point in the decision.

4. Work Assignment

The TWINS project and its participants have research needs for ALM and ALM tools and methods. There are also requirements to facilitate embedded testing. For embedded testing a testing framework, named Probe Framework, is built and it requires a database interface, called Probe DB. The work assignment consists of two separate needs: Probe DB development needs and Microsoft Team Foundation Server research needs. These needs can be met by using Team Foundation Server for Probe Database development project. This section describes the actual work assignment, and more specifically the Probe DB development needs and requirements.

4.1 Probe Database Development Needs

Probe Database needs to be developed to work as a part of a larger testing framework. Probe Database's role is to work as a database interface for an embedded testing solution named "Probe Framework". Probe Framework consists of Probe Network and Probe Database. Probe Network is an application which gathers various kinds of information in an embedded device during testing. Information gathered from probes needs to be saved to a database through the database interface, "Probe DB". Exporting saved data should also be supported. The software should be platform-independent which must be taken into consideration when deciding on the programming language and database solution.

Probe DB should be able to import data over a TCP connection or from a file. Data coming from embedded device probes can be in binary or CSV format. These formats must be designed in cooperation with Probe Framework developers and specification documents must be made. Binary format is important to reduce work load and resource needs at the embedded device under testing. CSV

format is useful in prototyping Probe DB and in the early phase of integration testing.

Probe DB should also have data export functionality. Data from embedded testing can be analyzed with third party software which needs data to be in a specific format. There must be a method to specify data which needs to be exported and it's formatting. This is going to be done with an XML file which user edits before export. The XML file must be designed and specification document should be made.

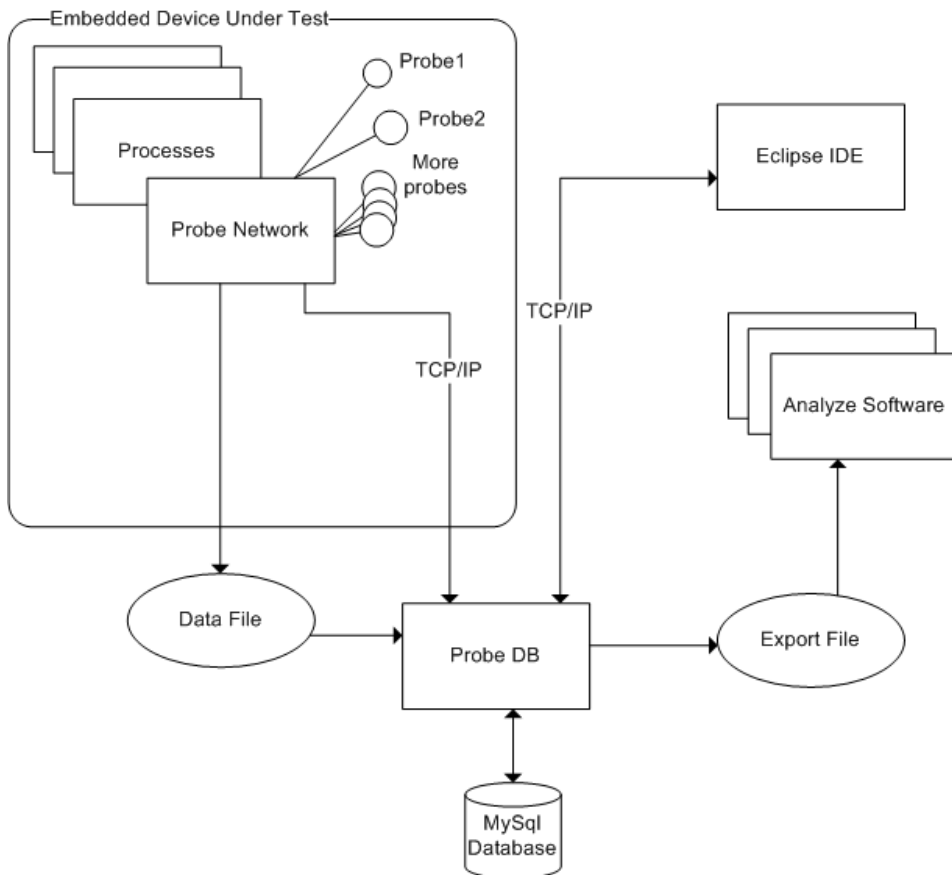


Figure 6. A graphical presentation of Probe Database's interfaces.

Other development needs for Probe DB are text based user interface, eclipse integration interface, reporting and data browsing. Probe DB is planned to be integrated with Eclipse, so an integration interface for it should be developed.

4. Work Assignment

Probe DB's main features, like import and export, could be used from Eclipse IDE which might be integrated with other useful tools like analysis software. Probe DB's export feature could also be directly integrated to some analysis software. For reporting features testers wished for some reports about test data saved to database. Browsing the data would also be handy in many situations.

As seen in Figure 6, Probe Database has TCP/IP interfaces to Probe Network and Eclipse. Test data is imported from the Probe Network over a TCP/IP connection or from a file. Eclipse can control and get information from Probe DB over a TCP/IP interface. Getting data to analyze software can be done with Probe DB's export feature which exports specified data in specified format to a file.

4.1.1 Functional Requirements

Probe DB should be able to import data from a Probe Network over a TCP connection or from a file to which Probe Network has saved its data gathered from testing of embedded device. Database export that would export data from a database specified in export XML file should also be possible.

4.1.2 Technical Requirements

There are many technical requirements and issues that had to be taken into consideration when developing Probe DB. Binary and CSV formats and export XML format specify a large part of these. Also the TCP/IP connection interface, Eclipse interface and database queries face some technical issues.

Technical requirements for import features are supporting import from a file and import over a TCP/IP connection. An import feature must be able to import data in binary format or CSV format. These have their technical requirements.

A binary format protocol should be designed to minimize the redundancy of data and facilitate generating data at the Probe Network running on embedded devices which are often quite limited in resources and computing power. Thus there are many technical requirements concerning binary import format. Binary format should be able to describe all necessary test information created by the Probe Network. The majority of test data is information coming from probes but there is also input data, different output types and data types. Test information always belongs to a specific project, version and test case, so there must be a way to specify this information. Data coming from the Probe Network can be in

various forms, which must be identified correctly. Binary format should support byte order since it may vary in different embedded platforms. There should also be support for all basic data types: Boolean, integer, float, text and binary. Input and output values are often just integers of different sizes, so binary format should have one, two, three, four and eight byte integers to support large numbers, and at the same time minimize the size required for saving small integers. Different time scales are often present in testing, ranging from a few clock cycles to even hours of testing, so there must be different time resolutions available. In the future new versions of binary protocol format might be developed so different protocol versions must be supported. To reduce redundancy there must be a way of sending multiple input or output data units in a group without repeating their header information.

Export feature includes XML format specification, which has been designed according to technical requirements of the export feature. The user must be able to specify data he wants from the database, its formatting, filtering conditions, data manipulation and so on. Data to be retrieved always belongs to a specific project, version and test case so this information must be specified in XML. Exported data is often output and input values, but it must be made possible to choose the database table from which the information is retrieved. Filtering data according to search conditions, limiting the number of results and getting data in a specific order must also be possible. For formatting data there must be a way to specify line start and separator characters and the order of data fields. Data manipulation features, such as scaling and replacing, are necessary to fit data for analyze software. Probe DB's export feature could have some export XML models which format data for known analysis software.

Database is required to store all information concerning embedded tests. This includes the project, version and test case information and input and output data. There is a need for output type and data type tables also.

4.2 MS TFS Research Needs

The TWINS project has research needs for Application Lifecycle Management. One goal of the project is to find proper tools and methods for project management and configuration management in distributed development project. Microsoft Team Foundation Server has been selected for the ALM tool to try different methods for ALM and to research how well it fulfils the tasks of an ALM tool. One project partner also has the more specific need of investigating the use of

4. Work Assignment

the SCRUM process model in TFS and bringing, at least partially, requirement management to it.

To investigate TFS as an ALM solution a demo project must be implemented. During the demo project experiences of the use of TFS are gathered and TFS' quality as an ALM is evaluated. Process model for the demo project will be modified SCRUM process with a requirement management item added to it. Thus a modified process template will also be validated during the demo project.

5. Implementation

In the implementation phase the research needs and Probe DB development needs are both going to be fulfilled during the same demo project.

5.1 Development Environment Setup

Before implementation the development environment had to be setup. For project management Microsoft Team Foundation Server and the necessary plug-ins to perform the customizations were installed. Visual Studio and the TFS client, Team Explorer, was setup for coders.

5.1.1 TFS Installation And Configuration

Team Foundation Server was installed on a Windows Server 2003 machine as a single server deployment. TFS Build was also installed on the same server machine.

Installation required many steps, but instructions provided on the Microsoft website were quite good. Before actual TFS installation other necessary software had to be installed. Other software TFS requires are IIS, ASP.NET, Microsoft SQL Server 2005, Microsoft SQL Server 2005 SP1, Microsoft .NET framework hotfix and SharePoint Services and its critical updates. IIS and SharePoint Services are needed for browser-based features like project and report portal. All data in TFS, project artefacts, source code, documents, etc. are stored to Microsoft SQL Server 2005 databases. Installation also required two domain accounts: a TFS Service account with local admin rights and a TFS Reporting account. The TFS Service account is used to run TFS services on a server machine and the reporting account to run SQL Server Reporting Services. Also, installation steps had to be done with a domain account which had local admin rights.

5. Implementation

Despite good instructions some mistakes were made during the first installation attempt. The first installation attempt was mistakenly done with a local administrator account and local service accounts. There were great difficulties getting SharePoint services to work correctly. Finally installation was successful, but using active directory accounts for user identification did not work. After reading instructions more carefully the server was re-installed and necessary domain accounts were used. This time installation went through quite easily without any bigger problems.

After successful installation some configuration was required and installation of the “Agile Software Development with SCRUM” process template, Team Explorer and Process Editor Visual Studio plug-ins. The “Agile Software Development with SCRUM” process template included its own installer which was run at the server machine. Process Editor plug-in was easily installed to Visual Studio and it needed little configuring.

5.1.2 Process Template Installation And Customizations

Installing new process templates to TFS is quite simple. Process templates can be managed with the Process Template Manager located in the Team menu. The SCRUM alliance’s “Agile Software Development with SCRUM – v1.2” had its own installer which was run at the server machine. For customizations, the Process Editor plug-in was also installed.

Customizing process templates requires some study. Process templates consist of XML process definition files, plug-ins and a new project wizard. XML process definition files define tasks which are run when creating a new team project. They also define work items, work item queries, areas, iterations, the project portal, version control permissions, check-in notes, report and groups and permissions. Plug-ins are components that run when a new project is created. They create necessary files and configure data for the project. Plug-ins read XML process definition files and do the necessary tasks defined there. The new team project wizard is run when new team project is created. It runs plug-ins and configures the team project according to XML process definition files.

In process templates there are six main XML process definition files, which are Work Item Tracking XML, Classification XML, Windows SharePoint Services XML, Version Control XML, Reports XML and Groups and Permissions XML. Work Item Tracking XML specifies work item types, work item queries and initial work items that are created when project is created. Each work item

has also its own XML definition file which defines its data fields and behavior. Classification XML defines areas and iterations. Areas are used to organize work in different areas, for example UI, database and applications. Iteration is a set of activities which are repeated in iterative software development. Work items are grouped by iterations. Windows SharePoint Services XML specifies which site template is used for the project portal and define which additional document libraries and folders and files are created there. Version Control XML defines initial version control permissions, check-in notes for project and whether multiple check-out option is enabled. Reports XML creates the reporting site (using a plug-in) and defines initial reports and report folders for the project. Groups and Permissions XML contains initial group and permission definitions for the team project.

Customizations to the process templates can be made by manually editing the process template's XML process definition files or with Visual Studio plug-in, Process Editor, which is available at Microsoft's website. The Process Editor plug-in facilitates process customizing by providing UI for showing and editing process templates. One does not have to understand XML to customize process templates with Process Editor.

The process template used in demo project was SCRUM alliance's "Agile Software Development with SCRUM – v1.2". Customizations were made using Process Editor. A new work item, called "Idea", was created to work as a requirement management item. It was made by first making a copy of Product Backlog Item and then modifying its data fields, layout and workflow. Idea work item's data fields are described in table 1.

The work item's data fields seen in Table 1 are used to contain and refer information of the work item. For example, fields in the work item layout require a unique data field. Data fields can also have rules, such as what values a field can have, or even conditional rules. Fields with RefName starting "System" are assigned and used by TFS and they are the same in every work item.

The Figure 7 presents 'idea' work item's workflow which describes the state transitions of the 'idea' work item. First when an idea is created its state is "Raw Idea", meaning it is not refined to a product or sprint backlog item, or taken into consideration in any way in implementation. When 'idea' is somehow taken into consideration, its state changes to "Used Idea". 'Idea' can also be deleted from states "Raw Idea" and "Used Idea". 'Idea' with the state "Deleted" can be revived to state "Raw Idea" again, however.

5. Implementation

Table 1. The data fields of an Idea work item.

Field name	Type	Ref Name	Description
Id	Integer	System.Id	Work item's i.d.
Title	String	System.Title	Work item's title
History	History	System.History	Needed to record work item's history
State	String	System.State	State of work item
Audit	String	System.VSTS.Scrum.Audit	Needed for work flow
Description	Plain Text	System.Description	Description of work item
Idea Type	String	VTT.VSTS.Scrum.IdeaType	Idea type: Feature, Technical Requirement
Idealist	String	VTT.VSTS.Scrum.Idealist	Idea creator
Importance	String	VTT.VSTS.Scrum.Importance	Importance: low, medium, high
Reason to Decision	String	VTT.VSTS.Scrum.ReasonToDecision	Reason to current state of idea

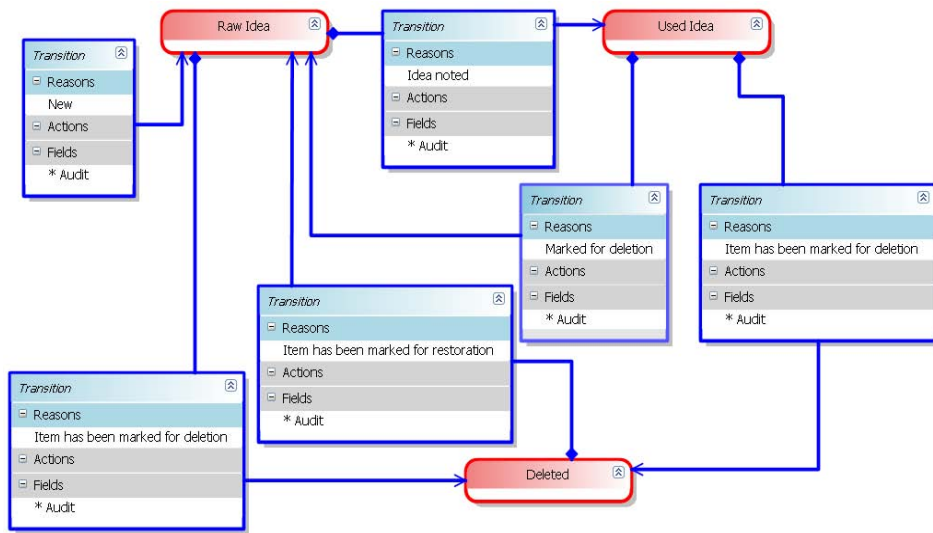


Figure 7. Workflow of idea.

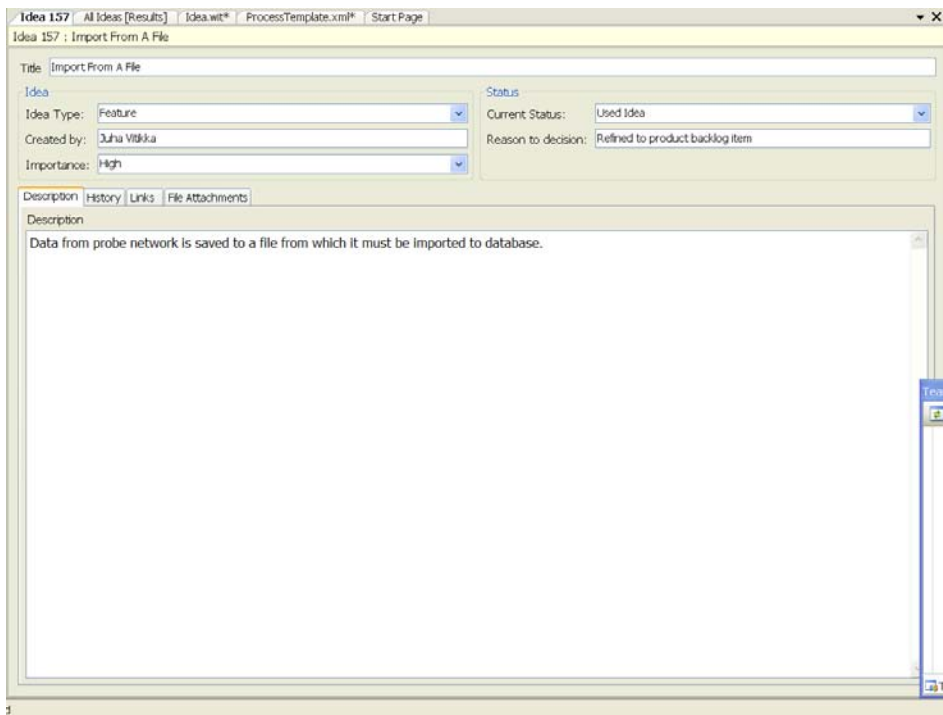


Figure 8. An idea work item in action.

5. Implementation

An 'idea' work item can be created from team explorer as any other work item. For creating and editing the 'idea' work item, TFS opens the 'idea' form shown in Figure 8. In addition to 'Title' it has the following fields: Idea Type, Created by, Importance, Current Status and Reason to Decision. 'Idea Type' is either a feature or technical requirement. Importance can have the values High, Medium or Low and Current Status Raw Idea, Used Idea or Deleted. 'Idea' also has the fields Description, History, Links and File Attachments which are included in every work item. In the description field the user can write the description and other additional information about the work item. The 'History' field shows change history of the work item. In the links field can be seen the work item's relations to other work items. Finally, to file attachment field can be attached documents or other files related to the work item.

The 'Idea' work item is meant to be used for requirement management. Requirements are gathered before the start of project and during the project they are updated as requirements change. Feature ideas are often taken to the product backlog, whereas technical requirement ideas might affect some technical solutions and decisions. All ideas are not necessarily taken notice of at all so they remain as raw ideas or are deleted at some point.

5.1.3 Project Setup

Before the demo project could be started some planning and preparations had to be done. The project team had to be gathered, working environments setup, requirements gathered and roles made clear.

Setting up the demo project was started by gathering requirements for Probe DB from the customer and Probe Network developer. Teemu Kanstren, who is working on his thesis at VTT acted as the customer, and the Probe Network is developed by Markku Pollari as a part of his master thesis. When a clear view of requirements was established, the project was planned according to SCRUM. The requirements were divided into logical groups and initially scheduled so that the working product would be delivered at the end of the project. The project also required one additional member to test distributed development with TFS. A TWINS project member, Juho Eskeli, volunteered to be a second contributor in the demo project. He does not actually do any coding or otherwise contribute to the Probe DB development but he acts as he would by marking tasks and bugs done. Juha Vitikka fulfils the role of project manager and contributor in the pro-

ject. Thus in addition to coding Probe DB, he takes care of coordinating the project and its work.

5.1.4 Visual Studio and Python Integration

Python was selected as the programming language for Probe DB but Visual Studio does not have native support for it. Visual Studio must be used to use TFS features in the project, so some way of integrating VS with Python had to be found. There is .NET implementation of Python called IronPython and Visual Studio SDK has a sample integration project of it.

IronPython syntax is almost the same as in native Python, although it misses some useful modules which needed to be added separately. The missing modules were taken from FePy, which is an enhanced implementation of IronPython. Some modules did not work by themselves, but required an additional .dll –file to offer support for some Python functions. In this project files “IronPython.dll” and “MySql.data.dll” were needed. IronPython.dll offers some Python system level functions and MySql.data.dll was required by mysqldb.py module.

5.2 Probe Database Development

Development was partially directed by SCRUM, since it was selected to be a process model for this project. It was decided to run the project in four sprints. Some designing was done before the initiation of the project. The architectural design of Probe Framework was drawn from the feature and technical requirements.

Probe DB has eleven classes, which can be seen in Figure 9. TUI, short for Text Based User Interface, is the main class that works as a user interface. It initiates functionality in other classes according to the user’s choice. When TUI starts, EclipseTCPInterface which listens to socket for incoming Eclipse control messages, is also launched to a separate thread. EclipseTCPInterface receives XML control messages and passes them to CommandParser. CommandParser parses commands, executes them and then returns the response back to EclipseTCPInterface which sends data to Eclipse IDE.

5. Implementation

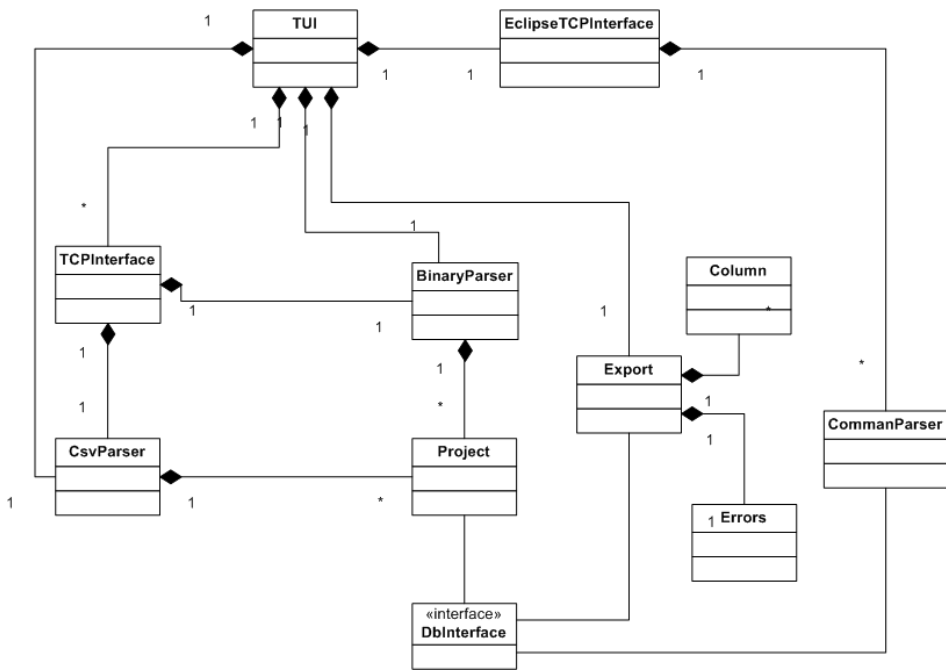


Figure 9. Class Diagram of Probe DB.

Import functionality works through TCPInterface, BinaryParser and CSVParser. In case of file import, TUI class initiates CsvParser or BinaryParser depending on the data format used. Import over network is taken care of by TCPInterface class. It starts to listen to the socket for incoming data, and depending on the data format, either passes it on to BinaryParser or CsvParser. These parsers parse information from formatted data and save it to the database through Project class. Project class has an instance for every test case that is importing data. Finally DbInterface module has the database queries to interface with database.

Probe DB development was started from making CSV parser and database interface. The aim during the first sprint was to deliver the first working build of Probe DB which would be able to parse data from CSV file to database. That build would be used only for testing, not released to the customer.

Work was started by making an initial design for database relations. The requirements for database relations changed on many occasions during the project so here only the final version of the database model is discussed. A diagram below shows the final database tables and structure.

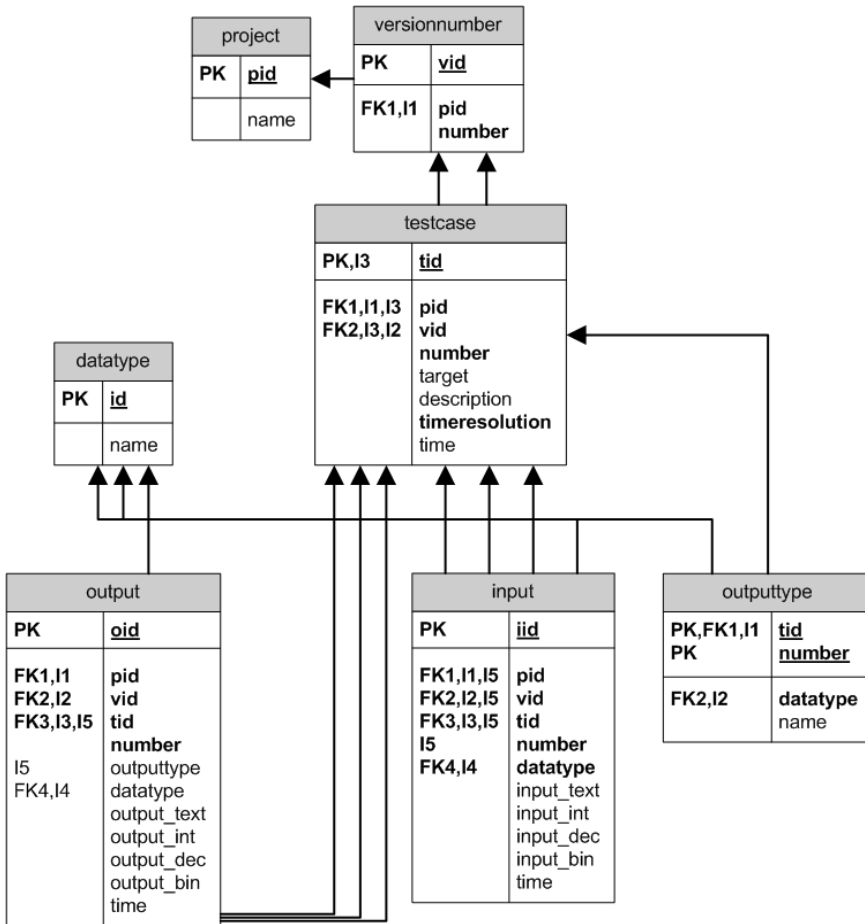


Figure 10. Database model of Probe DB.

Figure 10 shows Probe DB's database tables and their relations. There are five tables needed for storing all test data coming from Probe Network and in addition two auxiliary reference tables. All of the ID fields would not necessarily have to be included in tables but keeping them there keeps queries simple and probably makes them work faster.

All data imported to database must be connected to a specific project, version and test case, so there are tables for this data. The project name and version tables include only the name and version fields in addition to the ID fields. The table for test cases has ID fields and test case number, target, description, time resolution and time fields. Other fields are quite self explanatory but the time resolution field which tells what unit of time is measured in input and output

5. Implementation

times. The actual test data is saved to input and output tables. Input data is data that the Probe Network inputs to the system under test or values, configuration information, text or binary data used in tests. Output data can be various kinds of data coming from probes or the Probe Network itself. For example it can be test configuration data, memory dump, text, numbers of memory and CPU usage or test duration times. For different types of output data there's a reference table to which the output types, their data types and IDs are saved. Input and output data can be in different data types, so there is a reference table for data types as well. Thus, because of the data being in different data types, input and output tables have their own field for each MySQL data type. In addition, the output table has number and time fields and reference ID field to output type. The input table also has number and time fields and a reference ID field to data types table.

The CSV format was designed for the first working builds as a proof of concept, to help develop other parts of Probe DB and to identify more technical requirements and impediments. It was much easier to develop than binary format which is going to be used as the primary import format. When the CSV specification document had been written, coding of CSV parser and database queries was started.

Binary format and binary parser were developed with the aim of reducing work load at the probe network because it is usually running on an embedded device with little computing power and few resources. Also, the technical requirements had to be taken into consideration. Format protocol was formed of compact binary messages which have specific data fields. This is suitable in both cases, importing from a file and from TCP/IP packets. Most of the message fields are fixed length and for variable length fields, like text and binary fields, the lengths are given in the length field before the actual variable length field. Thus it is known how long each message is so messages can be put one after another without separator characters.

The message types in binary format are initial binary, output type binary, input binary, output binary and serial binary messages for sending multiple inputs and outputs in a row. Every data import must start with an initial binary message which delivers byte order, project name, version, test case, time and time resolution information. With the initial binary a three byte ID to reference the specific initial binary information is also sent. This is necessary because there can be multiple imports going on simultaneously. When importing data over TCP/IP every packet must start with this ID to ensure that the packet's data is matched with the right initial binary data. An output type binary message enables adding

custom output types to the database. It has fields name, number and data type number. For every test case there are automatically created output types “Test Result”, “Duration”, “Time” and “Configuration”.

Input and output binary messages are used to import the actual test data. Input binary includes input number, data type, time field and data field (with length field depending on data type). Output binary fields are output number, output type, time and the actual data field. For getting multiple inputs or outputs to the same message there are two kinds of serial binary types for inputs and outputs. First, the serial binary type enables the sending of multiple data and time pairs in the same message. An other serial binary type is for sending just multiple data values in the same message without the time field. The benefit from this is reducing redundancy by not sending other message fields with every data value.

Binary protocol format is processed by BinaryParser class’ instance. The instance is first given a file name or data string from TCP/IP packet, and depending on that it either reads data from a file or starts directly processing the data packet. Parsing itself is done by reading the binary message’s bytes at the pointer mark. The first byte of each message suggests what kind of message it is. Each message type is parsed and handled differently. If data fields are of fixed length the whole message can be parsed at once. In case there are variable length fields in the binary message the parsing must be done in parts because the places of data fields behind variable length fields are not known. When the whole message is parsed its information is processed and sent to an instance of Project class.

TCP interface creates a socket and then waits for incoming connections. Every socket is opened to its own thread to enable simultaneous multiple imports. Socket threads must also have their own instances of BinaryParser and CsvParser classes to parse data parallel in case it is coming from several connections. TCP interface receives packets of data and identifies whether data is in binary or CSV format. Then the data is given to the right parser. When parsing is finished the next data packet is taken to processing. After all data packets are received the socket is closed and the TCP interface keeps on listening to new connections.

Export functionality required designing of XML export format protocol which should be able to describe the information the user wants and its formatting. In export XML format all information is presented between XML elements. Below is an example of XML export file.

5. Implementation

```
<export>
  <bind>
    <project>Test</project>
    <versionnumber>alpha</versionnumber>
    <testcase>1.0</testcase>
  </bind>
  <output>
    <linestart>::</linestart>
    <separator>;</separator>
    <start>50</start>
    <max-values>20</max-values>
    <table>output</table>
    <column>
      <colname>time</colname>
    </column>
    <column>
      <colname>output_int</colname>
      <scale-max>7</scale-max>
      <scale-min>4</scale-min>
    </column>
    <column>
      <colname>number</colname>
    </column>
    <where>number>100</where>
    <order>number,output_int</order>
  </output>
</export>
```

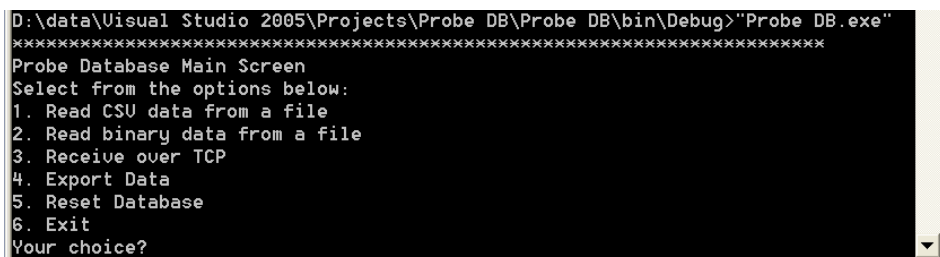
There are two main elements, `<bind>` and `<output>`. First there must be a described project, version and test case that the data to be exported belongs to. This is specified in the `<bind>` element. Within the next element, `<output>`, all the other information about the export is described: the table from which to export, columns taken to export, order, scaling, formatting and filtering options. Using the `<order>` element the user can choose a column by which the values are ordered in either ascending or descending order. Filtering elements enable the user to filter out a number of first rows, define the maximum number of rows and specify search parameters according to SQL syntax. For formatting export data there is a possibility to specify line start and separator characters. The scaling option is column specific and it enables specifying minimum and maximum that values are scaled to.

The XML export format parser was implemented in `Export` class using Python's `pyexpat` module. It is a very simple parser which offers the possibility to write methods that are run when the parser detects start element, data element

and end element. Export class also uses Errors Class and Column Class as auxiliary classes. Errors Class has a definition for one exception and instances of Column Class are used to contain information concerning single columns. There is no error checking for XML files so it is up to the user to write the XML correctly following the format specification.

Eclipse integration works over a TCP/IP connection. It is implemented in EclipseTCPInterface class and it offers Eclipse a way to retrieve data from the database. Integration interface is started on its own thread from TUI class. It creates a socket and listens to incoming connections. A connection is opened from Eclipse IDE and then the interface waits for data. When data arrives it first checks that the data, or at least its first element, is in XML format. The XML data is given to an instance of CommanParser class which parses XML using Python's pyxpath parser described earlier (XML export format). Eclipse interface offers queries for projects, versions and test cases. Later in future versions it may expose all Probe DB's functionality to Eclipse IDE.

Figure 11 represents the User interface of Probe DB. It was constructed quickly and is a quite simple text-based UI. It is implemented in TUI class which also works as the main instance of Probe DB. It shows the user which choices of functionalities Probe DB offers and then waits for the user to make his or her choice. When a choice is made, a specified feature starts to run and when it completes it, UI informs the user of the choices again. Below is a screenshot of UI main screen.



```

D:\data\Visual Studio 2005\Projects\Probe DB\Probe DB\bin\Debug>'Probe DB.exe'
*****
Probe Database Main Screen
Select from the options below:
1. Read CSU data from a file
2. Read binary data from a file
3. Receive over TCP
4. Export Data
5. Reset Database
6. Exit
Your choice?

```

Figure 11. Text based UI of Probe DB.

5.2.1 Testing

Visual Studio's IronPython integration does not offer support for automated tests which presented considerable difficulty in implementing tests. This was solved by implementing tests in Eclipse which has PyUnit offering support for unit tests. Unit tests were not written for all source code but the most important parts and complex parts have unit tests.

Several integration tests with Probe Network have been made. The first time integration with the Probe Network was tested when a working build with CSV parser and database file import functionality was ready. Probe Network developer made a sample CSV data file for testing import functionality. First integration testing revealed a number of bugs, but it also proved that the concept works, and gave some ideas for future development. Later more integration testing was done when the binary parser and network import features were ready. The binary parser was tested the same way as CSV parser integration. Network import was tested a few times with a small amount of sample data, but all of the technical requirements could not be tested since the Probe Network is still under development. Eclipse integration was also tested by Juho Eskeli while developing eclipse integration plug-in. Integration tests revealed many bugs, but after fixing them Probe DB proved to be working well with Eclipse and Probe Network as far as requirements were able to be tested. Binary format seemed to reduce the probe effect, that is mainly work load and need of resources, at the embedded device when compared to CSV format. Export functionality was tested by importing random data to the database and then making an XML export file for an MVA tool named Pervis developed at VTT. Data was exported and read successfully to Pervis.

5.2.2 Limitations

The software's UI is very clumsy and unattractive since it is text based. Probe DB does only one thing at a time (excluding Eclipse integration which runs in the background). You can not, for example, import and export simultaneously. The TCP/IP connection does not buffer packets, so there is a possibility that data gets lost when the data is coming in too fast. Eclipse integration does not yet offer support for all Probe DB features but more available features will be added in future development.

5.3 Software Development with TFS

TFS offers a wide range of tools and features facilitating the work of project managers especially but also work of other project members. All project members are able to get and share all necessary information using TFS or its project portal. The project is coordinated using work items.

5.3.1 Working with MS TFS

There are different roles in the project which all have different usages for TFS. Project managers coordinate the project and its work with the help of work items and reports. Coders use source control repository, project portal and work items to manage their work and schedule. Process templates have some automation for project flow and process guidance guiding through different stages of project. The project portal can be used to share essential information, documents and reports with group members.

From a project management point of view TFS has useful features for project coordination and information sharing. The project manager uses work items to coordinate project work and to inform other project contributors of what their tasks and schedules are. The project portal is used to share all kinds of essential information like documents, links, events and contacts. From reports, a project's state and progress can be monitored. Process models have also process guidance providing information about the process and its methodology, work procedures, roles, work items and reports.

In practice, the project managers' work with TFS is mainly playing with work items, observing the progress of work and watching reports. Getting started with a new project is quite simple if the process model is otherwise familiar. The process template takes care of creating the project and its project portal and reports. After the project is created it needs to be configured for the project's needs. This includes adding members to the project, setting up permissions and security, tailoring the project portal and configuring source control settings. The next phase is usually setting up project management, meaning creating the project's initial work items. Creating, viewing and modifying work items are done from Visual Studio Team Explorer. When the project gets going the project manager observes and coordinates project work with the help of work items and reports. Detailed work progress can be seen straight from work items as contributors mark them done, but the bigger picture can be seen from reports. They

5. Implementation

are available from Visual Studio Team Explorer plug-in and from the project's report portal.

For coders, TFS offers integrated source control repository and facilitates getting and sharing information and coordinating work and schedules. Coding itself is done with Visual Studio, which integrates with TFS. Source control files are managed with TFS' source control repository. From work items coders see requirements, issues, features and bugs of software they are developing and tasks that are to be done.

During project iteration coders select the tasks to work on from work items. They update work items according to their work progress so that project managers can keep on track of work. Work items can be linked to other work items, source control items and documents. This offers very useful traceability information for coders. For example, this way bugs can be traced to specific work items and source control files. For coders, the project portal works as a place for information sharing. There information can be shared using document libraries, discussion boards and links, events, announcements and contact lists.

The project portal greatly facilitates project communication. It is quite easily customizable for a projects' needs. There can be added reports, discussion boards, document libraries and various lists, like event, announcement contact, link and task lists. It is easily accessible with web browser.

5.3.2 Software Configuration Management

TFS has an integrated source control repository which uses SQL Server 2005 as information storage. In TFS the source control files are managed with check-in and check-out operations. Check-out takes files under editing and check-in commits the changes to repository. Control management in TFS is based on change set which describes one check-in operation and all files and other objects associated with it. TFS source control has support for the common control management operations, like Merge, Branch and Shelve.

When a coder starts to work with a source control object he performs a check-out operation. A check-out downloads the latest version of files from the repository to be edited and marks them checked-out. If multiple check-out option is enabled then a coder can decide whether other coders can perform check-outs or check-ins at the same time. There may be difficulties merging the changes made to the source control file that has been edited simultaneously by several coders so usually it would be wise not to allow concurrent check-outs. When the coder

has finished editing source code files he performs a check-in, which commits the changes made to checked-out files. Depending on check-in policies check-in can (or must) be associated with work items. [20, pp. 152–156]

In addition to basic operations like check-in and check-out TFS source control's features include labeling, branching, merging, shelving, comparing and locking. Labeling is a way to mark a specific set of file versions so that this particular set is retrievable later if necessary. This feature, also called as baselining, is used very often to manage different versions and branches of product. The branching feature means that you can make several branches of product that are developed differently from the same baseline. Merging, that is merging two branches, is the opposite. Shelving is one kind of check-in operation in which the changes are not committed to the original source code file but are shelved to possibly be used later. [20]

In the demo project carried out, we used this check-in policy and found out it was quite practical. This way it can be traced back to which files a particular work item is associated with, or vice versa.

5.3.3 SCRUM

Managing SCRUM in TFS is possible with Conchango's SCRUM for Team System process template. It has work items, reports and process support needed for SCRUM. Work items in the SCRUM process template are Product Backlog Item, Sprint Backlog Item, Sprint, Release and Sprint Retrospective item. There are also many useful reports like Delta Report, burndown charts and reports of specific work items. SCRUM process template lacks process automation but it has process guidance for process support. SCRUM's roles can be managed from TFS membership and security settings.

Different SCRUM process artefacts in the SCRUM process template appear as work items. There are five actual SCRUM work items and one work item, an idea, was added to the process template in this implementation to work as requirement management item. All items have Description, History and File Attachment fields and most of them have also Links field for traceability. SCRUM product backlog is constructed of product backlog items. They have all the necessary data fields needed to control it according to SCRUM. A product backlog item is addressed to a specific sprint and product area. For product backlog prioritization and scheduling there are Relative Value, Estimate and Work Remaining fields. Current Status field controls product backlog item's state from "Not

5. Implementation

Done” through “In Progress” to “Done” or “Deleted”. With product backlog item’s Owned by field item can be marked for a specific developer. A sprint work item defines one sprint and its parameters. In addition to sprint status there are fields for sprint capacity, release number and start and end dates. SCRUM tasks, bugs and impediments appear as sprint backlog items. As product backlog items they also have Sprint Number, Product Area, Status, Owned by, Estimate and Work Remaining fields. The backlog item-type field specifies whether the sprint backlog item is either task, bug or impediment. Sprint retrospective issues can be gathered to sprint retrospective items. Their parameters are Sprint Number, Retrospective Type, Team and Individual. Retrospective types are “What went well”, “What didn’t go so well” and “What we can do better”. There is also a work item for release with only Release Number and Current Status fields.

The SCRUM process template has many reports, most of which just list work items, but there are a few quite useful ones also. Sprint and the product burn-down chart reports show the progress of the project by viewing graphically how many of the work items have been done, and how much there is still to be done. From a burn down graph can be seen a burn down line, from which can be estimated when all the work will be done. Sprint and product cumulative flow reports are also handy for observing progress. They show a graphical diagram of what is done, in progress and what is not done. Delta report shows changes in product backlog. Sprint overview chart shows in sprint’s tasks’, bugs’ and impediments’ remaining work in a graph.

The project was run successfully with SCRUM following its process and procedures. SCRUM meetings were only short formal events where the project leader (and coder) Juha Vitikka told Juho Eskeli which tasks he should mark done. These meeting were handy for gathering experiences of SCRUM and its procedures though. Requirement management item customization for TFS SCRUM template was used during the project and it proved to work as it should. Reports and queries of this item would have been useful, and they will probably be implemented in the future.

6. Discussion

In this section issues concerning this work overall and issues raised while working will be discussed. User experiences of TFS and SCRUM by the viewpoints of different roles were gathered during the project. Both the positive and negative issues and possible enhancements are discussed here. Also issues of database interface development are covered.

6.1 TFS Usage Experiences

TFS usage was tested with a two member team project. One purpose of it was to get user experiences of TFS in different roles. Roles in the project were coder, project manager, customer, user and ScrumMaster. Thus a large portion of TFS main functionality was used in the project.

TFS work items were found to be quite handy and simple for managing the project. Work items are mainly used by the project manager and coders. Though they lack some automation, they are also quit fast and easily manageable. For an automation point of view there could be more automatically updating fields. All information of work items are controlled by editing the data field which may feel a little frustrating in the long run. For example, tasks could be marked ‘done’ with a button or some other way. There is also currently no way to update information to multiple work items at the same time. For listing work items there could be dynamic queries which would list only those work items that are essential at the time. For example, task query lists all the project’s tasks, though old tasks are seldom used or observed. Controlling work items by hierarchy view would also facilitate controlling projects with a large number of work items. Work item control is integrated with Microsoft Project and Microsoft Office but in our demo project we found little use for these tools. If the project manager is more familiar with MS Project this integration probably facilitates his work.

For requirement management the ‘idea’ work item was used in the demo project. Though a slightly simplified implementation it worked quite well and provided useful traceability information between requirements and other work items. The ‘Idea’ work item may be improved in the future development.

TFS is used from Visual Studio with Team Explorer plug-in. Visual Studio is limited in programming language support so if you want to code java, for example, it is not easily done with TFS. It is possible to make other programming IDEs to integrate with TFS though. For eclipse IDE there is a commercial TeamPrise plug-in which enables the use of TFS from Eclipse.

The project portal was found to be a very practical tool for project communication and information sharing. It is easy to learn to use and it can be customized for a project’s needs.

All in all, ALM has provided some useful help in the demo project. Especially work coordination and communication with team members was somewhat facilitated by the use of TFS. One must notice that the demo project had only three member team and the product developed was not extremely complex. ALM and ALM tools might be more helpful in larger projects with more people and more complex products. In distributed software development it could also appear to be very important.

6.2 Experiences about Database Interface Development

Many issues must be taken notice of when developing a database interface for an embedded test framework. One of the most difficult issues was that data must be transferred between the embedded platform and database interface using transfer techniques and data formats that would not put too much work load on an embedded device. Thus a compact and efficient binary format had to be designed and tuned for optimal performance at the Probe Network’s end. The information saved to the database is analyzed with third party analysis software, so there had to be a customizable export feature also. For a database solution there were many possible options. MySQL was selected because it is free open source database, very commonly used and it supports SQL language. Other database solutions were not tried but an object oriented database was under consideration. MySQL was also a little easier to implement for Probe DB since the developer had no former experience of object oriented databases.

6.3 Probe Database Future Development

Planning for future development of Probe Database is already ongoing. Probe DB is going to be integrated with Merlin tool chain, which is a set of project management and testing tools integrated with Eclipse designed to facilitate distributed software development work. There is also an MVA (Multi Variable Analysis) tool that is going to be integrated with tool chain to Eclipse IDE and Probe DB. This puts development needs on Probe DB requiring its own ready-made export format and some automation. It is also possible that only the Probe DB's export module is developed to serve as Eclipse interface to Probe DB's database.

Some simple reporting features could be developed to offer testers some commonly needed information of test data. From quite early phases a report feature was considered for Probe DB but it was kept at low priority. Reports could be web browser based reports built using JavaScript or PHP with direct database connection. There could also be a web portal for reports. Reports themselves have not yet been planned but at least some basic ones which would show test cases and their statistics would probably be useful.

The User interface is quite clumsy and poor looking since it is only text based. If Probe DB is going to be used more commonly as a stand alone solution, not through eclipse integration, it would need a better UI. A graphical UI would look better and probably facilitate working with Probe DB.

Probe DB still lacks all information browsing features. Test case browsing could be added to UI, preferably to graphical UI, allowing the user to browse through projects, versions and test cases and maybe even the test data itself. Further on, with this test case browser user could maybe select some readymade exports for test data to some commonly used analyze software formats. Information browsing could be real time, meaning that the user could browse through information while the test is still running. An exporting feature might also be able to support real time export so that data could be analyzed real time while tests are running. Real time export could be sent directly to a network socket that is listened to by analyzing software instead of an export file.

7. Conclusion

During this work a working database interface, Probe DB, has been developed for an embedded testing framework, Probe Framework. Also a modification to MS TFS SCRUM process template was made and demoed. Probe DB was developed with the help of Microsoft's ALM solution, MS TFS, using a modified SCRUM process template. The project was carried out with SCRUM process model. The project provided insight to ALM and experiences of applying ALM in practice.

Probe DB was developed using the carefully selected methods and techniques described in this work. Because one major requirement was that the product would be open source and platform-independent it was coded with Python programming language. For the same reason MySQL database was selected to serve as the database solution. To avoid, or at least reduce, the probe effect Probe Framework's binary format was designed. It was made as compact as possible to reduce the need of resources and processing power at the Probe Network, which runs tests at the embedded platform. CSV format was designed before binary format to test the concept, and it is also needed if text-coded CSV information is wanted to be imported to the database from other sources. There was a need to analyze information with third party software, so a customizable export feature had to be developed. This was implemented using XML to describe the data and possible data manipulations wanted, and the format it should be exported in. File and network imports were required because embedded platforms and test setups vary so that both data transport methods are needed sometimes.

MS TFS with the modified SCRUM process template was successfully used to manage the demo project. MS TFS lacks requirement management items, so the SCRUM process template had to be tailored by adding an 'idea' work item to it. The 'idea' work item acts as a simple requirement item providing traceability and reporting possibilities for requirements.

References

- [1] Schwaber, C. (2006) The Changing Face of Application Life-Cycle Management. Forrester Research Inc. White paper.
- [2] Shaw, K. (2007) Application lifecycle management for the enterprise. Serena Software, April 2007. (Available 31.05.2007.) URL: http://www.serena.com/Docs/Repository/company/Serena_ALM_2.0_For_t.pdf
- [3] Doyle, C. (2007) The importance of ALM for aerospace and defence (A&D). Embedded System Engineering (ESE magazine), Volume 15, Issue 5, pp. 28–29.
- [4] Doyle, C. & Lloyd, R. (2007) Application lifecycle management in embedded systems engineering. Embedded System Engineering (ESE magazine), Volume 15, Issue 2, pp. 24–25.
- [5] Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002) Agile software development methods. Review and analysis. Espoo: VTT Publications 478. 107 p.
- [6] Royce, W. W. (1970) Managing the Development of Large Software Systems. Proceedings of IEEE WESCON 26, August. Pp. 1–9.
- [7] Pressman, R. S. (1987) Software Engineering: A Practitioner's Approach. McGraw–Hill Book Company. 161 p.
- [8] SCRUM Alliance Home Pages. (Available 30.4.2008.) URL: <http://www.scrumalliance.org/>
- [9] Kääriäinen, J. & Välimäki, A. (2008) Impact of Application Lifecycle Management – a Case Study. In: Proceedings of International Conference on Interoperability of Enterprise, Software and Applications, March 25–28, Berlin, German. Enterprise Interoperability III – New Challenges and Industrial Approaches. Mertins, K., Ruggaber, R., Popplewell, K. & Xu, X. (Eds). Springer. Pp. 55–67.
- [10] Eclipse home page. (Available 30.4.2008.) URL: <http://www.eclipse.org/>
- [11] Semeniuk, J. & Danner, M. (2005) Managing Projects With Visual Studio Team System. Microsoft. 249 p.
- [12] Hamou-Lhadj, A., Lethbridge, T. C. & Fu, L. (2004) Challenges and Requirements for an Effective Trace Exploration Tool. Proceedings of the 12th IEEE International Workshop on Program Comprehension (IWPC'04), 24–26 June. Pp. 70–78.

- [13] Sääksvuori, A. & Immonen, A. (2002) Product Lifecycle Management. Springer 2004 (Original Finnish version: Talentum, 2002.) 231 p.
- [14] Gait, J. (1986) A probe effect in concurrent programs. *Software–Practice & Experience*, Volume 16, Issue 3, pp. 225–233.
- [15] Ernst, M. D., Perkins, J. H., Guo, P. J., McCamant, S. Pacheco, C., Tschantz, M. S. & Xiao, C. (2006) The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*. Volume 69, Issue 1–3, pp. 35–45.
- [16] MySQL Web Pages. (Available 30.4.2008). URL: <http://www.mysql.com/>
- [17] Zope Web Pages. (Available 30.4.2008.) URL: <http://www.zope.org/>
- [18] Elmasri, R. & Navathe, S. B. (2000) *Fundamentals of database systems*. Third edition. Addison–Wesley. 1 000 p.
- [19] Garcia-Molina, H., Ullman, J. D. & Widom, J. (2002) *Database Systems: The Complete Book*. Prentice Hall. 1 152 p.
- [20] Guckenheimer, S. (2006) *Software Engineering with Microsoft Visual Studio Team System*. Addison–Wesley. 256 p.



Series title, number and
report code of publication

VTT Publications 714
VTT-PUBS-714

Author(s) Juha Vitikka		
Title Supporting Database Interface Development with Application Lifecycle Management Solution		
Abstract <p>Controlling a software project has a major effect on the project's productivity, expenses and the quality of a project's product and code. This work investigates Application Lifecycle Management which considers how software, the software process and its different phases are controlled. With the help of Application Lifecycle Management a working database interface for embedded testing framework has been developed.</p> <p>In this Microsoft Team Foundation Server is used for managing the software project. As a software process SCRUM is used by utilizing SCRUM for Team Sys-tem process template developed by Conchango. The process template is custom-ized to bring support for requirement management in to it. The customized proc-ess template is used in the demo project, in which a database interface for em-bedded testing framework is developed. Thus the process template customiza-tion is tested in practice and experiences of ALM, Microsoft TFS and SCRUM process are gathered.</p> <p>During development of the embedded testing framework, which is one sort of generic data gather-ing tool, one must pay attention to many issues such as data transport methods and formats, data-base solutions, data export methods and integration. Database interface software, called Probe DB, is developed accord-ing to the requirements of the customer. Own CSV and binary data transport formats and XML format for export functionality are designed, and interfaces for file and TCP/IP import and for Eclipse IDE are developed. Software is coded with Python and MySQL will serve as database solution.</p>		
ISBN 978-951-38-7353-0 (URL: http://www.vtt.fi/publications/index.jsp)		
Series title and ISSN VTT Publications 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)		Project number 6086
Date September 2009	Language English, Finnish abstr.	Pages 54 p.
Name of project TWINS		Commissioned by ITEA
Keywords ALM, SCRUM, TFS, data transport formats		Publisher VTT Technical Research Centre of Finland P.O.Box 1000, FI-02044 VTT, Finland Phone internat. +358 20 722 4520 Fax +358 20 722 4374



Tekijä(t) Juha Vitikka		
Nimeke Tietokantarajapinnan kehittäminen ohjelmiston elinkaaren hallinnan avulla		
Tiivistelmä <p>Ohjelmistoprojektin hallinta vaikuttaa merkittävästi projektin tuotteen ja koodin laatuun, tuottavuuteen ja kuluihin. Työssä perehdyttiin ohjelmiston elinkaaren hallintaan (ALM), joka käsittelee sitä, miten ohjelmistoa, ohjelmistoprosessia ja sen eri vaiheita hallitaan. Ohjelmiston elinkaaren hallintaa apuna käyttäen kehitettiin toimiva tietokantarajapinta sulautettuun testauskehikkoon.</p> <p>Tässä työssä ohjelmistoprojektin hallintaan käytettiin Microsoftin ohjelmiston elinkaarenhallintatyökalua Team Foundation Serveriä. Ohjelmistoprosessina käytettiin SCRUMia Conchangan TFS:ään kehittämän SCRUM for Team System -prosessipohjan avulla. Prosessipohjaa muokattiin vaatimustenhallinnan mukaan tuomiseksi. Muokattua prosessimallia sovellettiin projektissa, jossa kehitettiin tietokantarajapinta sulautettua testauskehikkoa varten. Näin testattiin prosessimallin muokkauksen soveltuvuus ja samalla kerättiin kokemuksia ohjelmiston elinkaaren hallinnasta, MS TFS:stä ja SCRUM-prosessista.</p> <p>Sulautettu testauskehikko voidaan katsoa generiseksi tiedonkeruutyökaluksi, jollaiseen tietokantarajapintaa kehittäessä pitää ottaa huomioon useita seikkoja, kuten tiedonsiirtomenetelmät ja formaatit, tietokantaratkaisu, tiedon vieminen sekä integrointi. Tietokantarajapintasovellus, nimeltään Probe DB, kehitettiin asiakkaan tarpeiden pohjalta. Sovellukselle suunniteltiin omat CSV- ja binäärimuotoiset tiedonsiirtoformaatit sekä XML-formaatti ulosvietävän tiedon määrittämiseen ja kehitettiin tiedosto- ja TCP/IP-rajapinnat testauskehikkoon sekä TCP/IP-rajapinta Eclipse IDEen. Sovellus koodattiin Pythonilla ja tietokantaratkaisuksi valittiin MySQL. Kehitystyön tuloksena syntyi toimiva tietokantarajapinta osaksi sulautettua testaustyökalua, Probe Frameworkiä.</p>		
ISBN 978-951-38-7353-0 (URL: http://www.vtt.fi/publications/index.jsp)		
Avainnimeke ja ISSN VTT Publications 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)	Projektinumero 6086	
Julkaisu-aika Syyskuu 2009	Kieli Englanti, suom. tiiv.	Sivuja 54 s.
Projektin nimi TWINS	Toimeksiantaja(t) ITEA	
Avainsanat ALM, SCRUM, TFS, data transport formats	Julkaisija VTT PL 1000, 02044 VTT Puh. 020 722 4520 Faksi 020 722 4374	

VTT Publications

- 696 Suvi T. Häkkinen. A functional genomics approach to the study of alkaloid biosynthesis and metabolism in *Nicotiana tabacum* and *Hyoscyamus muticus* cell cultures. 2008. 90 p. + app. 49 p.
- 697 Riitta Partanen. Mobility and oxidative stability in plasticised food matrices. The role of water. 2008. 92 p. + app. 43 p.
- 698 Mikko Karppinen. High bit-rate optical interconnects on printed wiring board. Micro-optics and hybrid integration. 2008. 162 p.
- 699 Frej Wasastjerna. Using MCNP for fusion neutronics. 2008. 68 p. + app. 136 p.
- 700 Teemu Reiman, Elina Pietikäinen & Pia Oedewald. Turvallisuuskulttuuri. Teoria ja arviointi. 2008. 106 s.
- 701 Pekka Pursula. Analysis and Design of UHF and Millimetre Wave Radio Frequency Identification. 2008. 82 p. + app. 51 p.
- 702 Leena Korkiala-Tanttu. Calculation method for permanent deformation of unbound pavement materials. 2008. 92 p. + app. 84 p.
- 703 Lauri Kurki & Ralf Marbach. Radiative transfer studies and Next-Generation NIR probe prototype. 2009. 43 p.
- 704 Anne Heikkilä. Multipoint-NIR-measurements in pharmaceutical powder applications. 2008. 60 p.
- 705 Eila Ovaska, András Balogh, Sergio Campos, Adrian Noguero, András Pataricza, Kari Tiensyrjä & Josetxo Vicedo. Model and Quality Driven Embedded Systems Engineering. 2009. 208 p.
- 706 Strength of European timber. Part 1. Analysis of growth areas based on existing test results. Ed. by Alpo Ranta-Maunus. 2009. 105 p. + app. 63 p.
- 707 Miikka Ermes. Methods for the Classification of Biosignals Applied to the Detection of Epileptiform Waveforms and to the Recognition of Physical Activity. 2009. 77 p. + app. 69 p.
- 708 Satu Innamaa. Short-term prediction of traffic flow status for online driver information. 2009. 79 p. + app. 90 p.
- 709 Seppo Karttunen & Markus Nora (eds.). Fusion yearbook 2008. 2009. Annual report of Association Euratom-Tekes. 132 p.
- 710 Salla Lind. Accident sources in industrial maintenance operations. Proposals for identification, modelling and management of accident risks. 2009. 105 p. + app. 67 p.
- 711 Mari Nyssönen. Functional genes and gene array analysis as tools for monitoring hydrocarbon biodegradation. 2009. 86 p. + app. 59 p.
- 712 Antti Laiho. Electromechanical modelling and active control of flexural rotor vibration in cage rotor electrical machines. 2009. 91 p. + app. 84 p.
- 714 Juha Vitikka. Supporting database interface development with application lifecycle management solution. 2009. 54 p.