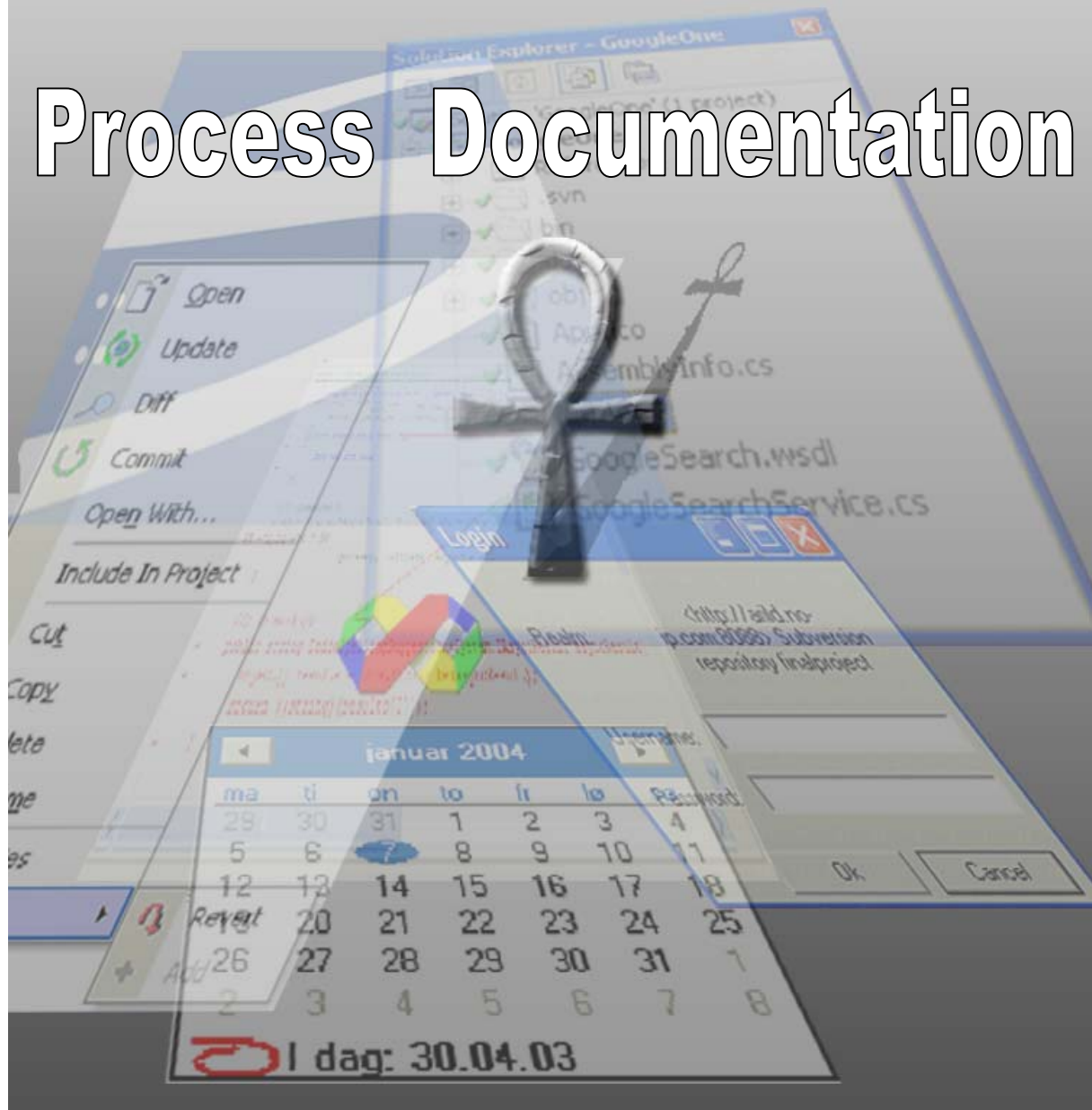


Process Documentation



Preface

This document describes how the process evolved. Challenges we encountered during the development phases, the reasoning behind the GUI implementation and the alternative approaches we considered are described in this report. The primary audience for this report is CollabNet and HIO. A dictionary containing the terminology used in this report is enclosed in the appendix.

Details about how the final product is constructed, the functionality and how to install Ankh is described in the product documentation. Tests performed on Ankh are described in detail in the test report.

The project group wanted to do a project using the Microsoft's .NET environment. One of the group members had used Subversion for several school and private projects during the last year and had the original idea for the project. By developing Ankh we hoped to deliver the useful Subversion tool to a wider audience for no cost in an open source community.

After having defined the project we contacted CollabNet through the Subversion developer mailing list, and they immediately showed interest in being the employer for our project and provided our external supervisor, Karl Fogel.

This report is optimised for a paper-version.

Thanks to

Eva Hadler Vihovde, HIO supervisor, for useful comments and guidelines for how a final project should be performed and an always-optimistic attitude.

Karl Fogel, external supervisor from CollabNet, for giving us the liberty to implement Ankh as we saw fit.

Ann Mari Torvatn, HIO for always being interested in our project and for dedicating so much of her valuable time to providing guidance for our documentation and visual presentation.

Thanks to all of you who have contributed to the testing of Ankh.

Table of contents

PREFACE	2
TABLE OF CONTENTS	3
1 INTRODUCTION	5
1.1 THE BIRTH OF THE PROJECT	5
1.2 REVISION CONTROL	5
1.3 COLLABNET	6
1.4 SUBVERSION	7
1.5 AIM.....	7
1.6 WHY USE ANKH?	7
1.7 WHAT CAN ANKH DO FOR YOU?.....	8
1.8 PROJECT SPECIFICATIONS	8
2 PLANNING AND METHODOLOGY.....	9
2.1 EVOLUTION IN THE GROUP PROCESS	9
2.2 ORGANISING AN OPTIMAL PROCESS ENVIRONMENT	10
2.2.1 <i>Commit mail</i>	11
2.2.2 <i>Build mail</i>	12
2.2.3 <i>Diary and Internet sites</i>	12
2.2.4 <i>Issue-tracker</i>	13
2.2.5 <i>Relations to the employer CollabNet</i>	13
DEVELOPMENT PROCESS	13
2.3 PILOT PROJECT	14
2.4 LEARNING PHASE AND TOOLS USED	14
2.5 DESIGN PHASE	15
2.5.1 <i>System design</i>	15
2.5.2 <i>Analysis of the users of our program</i>	16
2.5.3 <i>Mental model</i>	16
2.5.4 <i>Navigation model and menus</i>	17
2.6 IMPLEMENTATION PHASE	18
2.6.1 <i>General project challenges</i>	19
2.6.2 <i>NSvn wrapper layer</i>	19
2.6.3 <i>Integration of status icons in Solution Explorer</i>	20
2.6.4 <i>Ankh integration of menu options into VS .NET</i>	22
2.6.5 <i>Integration of Diff in VS.NET</i>	25
2.6.6 <i>Integration of Commit in VS.NET</i>	26
2.6.7 <i>Integration of file tree view of repository in VS.NET</i>	26
2.6.8 <i>Examples of reuse of user controls in the GUI layer</i>	28
2.6.9 <i>Ankh Help integration</i>	29
2.7 TESTING PHASE	29
2.8 DOCUMENTATION.....	30
2.9 DEVIATIONS FROM PROJECT SPECIFICATIONS	30
3 CONCLUSION	31
3.1 EVALUATION OF THE PRODUCT	31

3.2	EVALUATION OF THE PROCESS	32
4	REFERENCES	33
5	APPENDIX	34
5.1	DICTIONARY	34
5.2	DEVELOPMENT TOOLS	36
5.2.1	<i>VS.NET</i>	36
5.2.2	<i>VS.NET addin architecture</i>	36
5.2.3	<i>NUnit</i>	36
5.2.4	<i>NAnt</i>	36
5.2.5	<i>Python</i>	37
5.2.6	<i>C#</i>	37
5.2.7	<i>Managed C++</i>	37
5.2.8	<i>Enterprise Architect</i>	37
5.3	AVOIDING DUPLICATION OF CODE LINES IN GUI	37

1 Introduction

This chapter describes the background of the project, why Ankh should be used, what Ankh can do for you and the project specifications.

1.1 The birth of the project

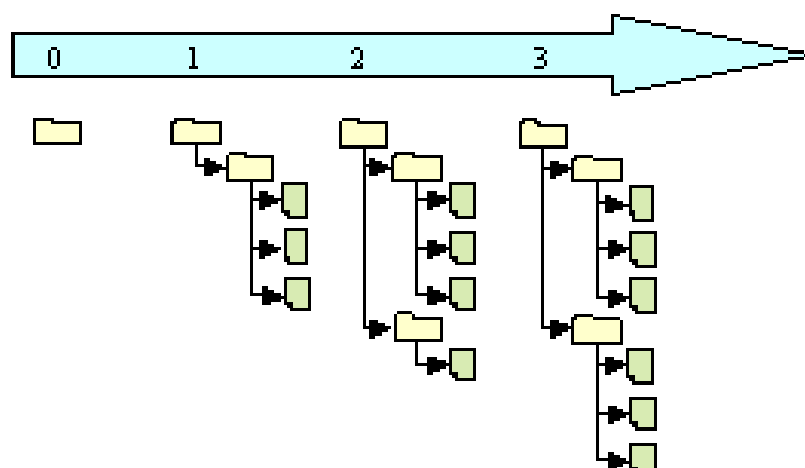
One of the group members had used Subversion for several school and private projects during the last year and had the original idea for the project. By developing Ankh we hoped to deliver the useful Subversion tool to a wider audience for no cost in an open source community.

Ankh is an Egyptian symbol representing both physical and eternal life. Based on the feedback we have experienced through the project period we know Ankh will be used.

1.2 Revision Control

In a typical software development environment, many developers will work on the same code base. Revision control makes it easier for these developers to share the same collection of files and directories in this code base. A revision control system is a centralised system for sharing this information where the core is a repository.

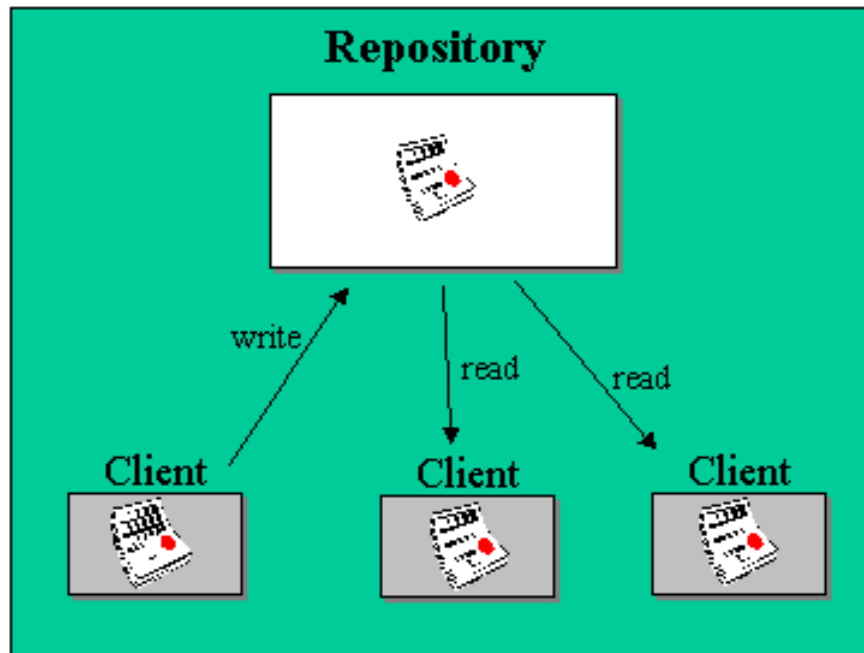
A **repository** is much like an ordinary file server, except that it holds every change ever made to a set of files. In the figure below yellow boxes indicates directories and green boxes indicate files. The numbers on the blue arrow indicate version numbers.



Each individual programmer checks out a working copy. A **working copy** is a collection of files and directories under revision control. Each file tree in the figure above is an example of a working copy that could be checked out from the repository. During the development work cycle these working copies are updated and synchronised with the repository.

When a commit is performed your changes are sent from your working copy to the repository. A **commit** makes your changes available to your colleagues. When a commit is performed it is customary to write a log message describing the changes made to the files.

Using revision control makes it possible to get an amazing overview of the work performed at any time from anywhere. If one project member has a problem that is difficult to solve another project member can easily get information about what was changed and support the other project member in solving the problem.



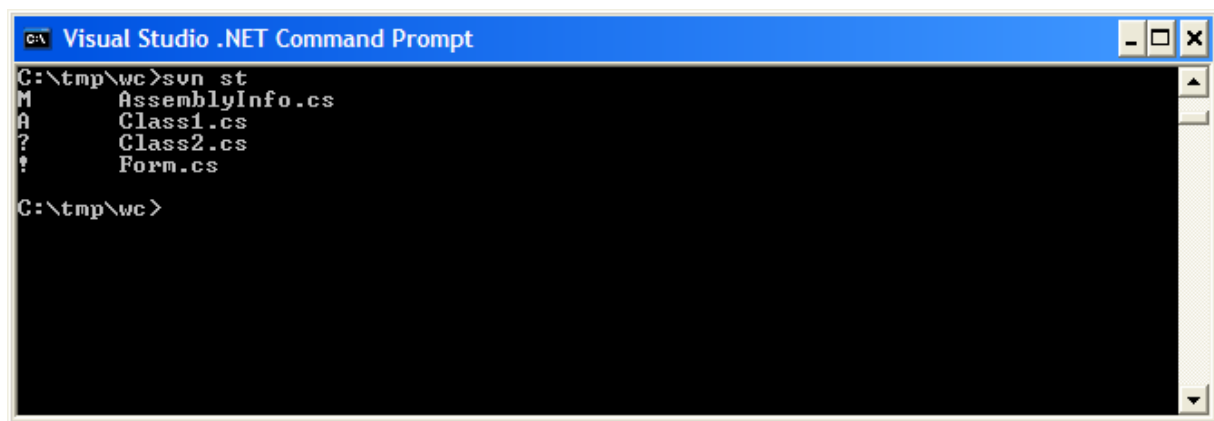
1.3 CollabNet

CollabNet is a California-based company that in the year 2000 provided the initial funding to begin development work on Subversion. CollabNet currently has several paid developers working on the Subversion project. They offer a wide range of products and services to improve software quality and accelerate time-to-market during product development. Their main product is SourceCast, which is a Web-based software development platform. Subversion aims to be the successor to an existing revision control system called CVS (Concurrent Version System) and CollabNet intends to make it a part of the SourceCast platform.

1.4 Subversion

Subversion is a free open source revision control system, which makes it easier for several people to work within the same set of files. The files are placed into a central repository. Subversion has a number of options. An attempt to analyse the Ankh implementation of these SVN options from the users point of view is done in the mental model in this report. The Ankh integration of these SVN options are analysed from the developers point of view in a conceptual model in the product report. Additional information concerning Subversion can be obtained from <http://subversion.tigris.org>.

When we started to work with Ankh Subversion (SVN) had to be used from the command prompt. The prompt below shows the status command “svn st” followed by the output from SVN.



```
C:\ Visual Studio .NET Command Prompt
C:\tmp\wc>svn st
M      AssemblyInfo.cs
A      Class1.cs
?      Class2.cs
?      Form.cs
C:\tmp\wc>
```

1.5 Aim

The aim of this project was to provide an integration for Subversion into the existing development environment Visual Studio .NET. By making an addin for Visual Studio .NET Subversion is made available to a huge new user base. It also eases the way developers work with revision control, since subversion commands are accessible directly from Visual Studio. The .NET wrapper layer is also available as a standalone component so that it can easily be used in other applications in the future.

1.6 Why use Ankh?

Ankh is a free open source tool that integrates the Subversion version control system into VS.NET. Currently there are several version control systems available that plug into VS.NET on the market, but few of them are free.

Subversion is the successor to a widely used revision control system named CVS (Concurrent Versions System). Subversion was initially founded to improve CVS. The initial goal of the Subversion project is to preserve the same core functionality as CVS and fix the most obvious flaws. Ankh is recommended because it is a useful free tool, which offers an improvement over the widely used CVS.

1.7 What can Ankh do for you?

When we started to integrate Ankh into VS.NET we decided to prioritise those options that were most frequently used on a day-to-day basis. We aimed to get a product that could be used as soon as possible. Ankh currently contains the following functionality:

- View status icons of a working copy in VS Solution Explorer (Status icons in Solution Explorer)
- Add files, directories, project and solutions (item) to the working copy (Add)
- Update working copy item to include changes from last revision (Update)
- Examine changes of item in working copy (Diff)
- Revert changes in item in working copy (Revert)
- Delete/remove item from working copy (Delete)
- Commit changes to the repository (Commit)
- Show file tree view of repository in Repos Explorer (Repository Explorer)
- Authenticate the user if the repository used is restricted

Ankh mainly supports Visual Basic (VB), Visual C# (VC#) and Visual C++ (VC++) projects. Support for VC++ projects is currently somewhat weaker than for VC# and VB projects. Details concerning the functionality of Ankh are described in the product documentation.

1.8 Project specifications

To ensure the sanity and quality of the system at any given moment, and in order to catch regression bugs, we conducted continuous unit tests using the NUnit testing framework.

We have provided NAnt build scripts that enable NSvn to be built without requiring the user to have Visual Studio.NET installed. The .NET wrapper layer is then available as a standalone unit for use in other applications.

The .NET wrapper layer covers the SVN client's API based on the last version of SVN.

Common programming guidelines were composed to get an overall consistent look of the code.

Subversion was used for revision control both for the source code and documentation. All documentation was written in English.

The group had one available PC at HIO. We installed the following to this computer:

- Windows 2000 5.00.2195 Service Pack 3
- Visual Studio .NET 7.0
- Subversion.

As CollabNet is situated in USA all our communication was conducted via e-mail. CollabNet did not provide us with any software or computer equipment.

2 Planning and methodology

This section describes how we worked through the different phases of the project in general.

2.1 Evolution in the group process

In the beginning of the project the group members worked together for shorter or longer periods almost every day until the whole group mastered all the new software and tools that were to be used for the project. In this period we worked mainly together- the three of us at the school computer and over remote sessions to computers located at our respective homes.

We started to work individually when all group members had successfully installed required software and tools on their home computers. We continued to meet at school also after we started to work independently at our home computers. In periods when we worked separately we communicated via MSN Messenger, e-mail and cell phone.

We intended to give all group members the possibility to participate in as many of the different aspects of the project as possible. All group members have coded in Managed C++, we have all made NUnit tests, and we have all coded C# in the GUI layer and the wrapper layer. All group members proofread all documentation.

During the project period we have discussed what to do and individually proposed what to work with. We have not had any group leader that has assigned work, instead we have taken the initiative for different tasks ourselves.

We decided early on that Arild was to be the contact person towards CollabNet, Per served as contact person for the computer department at HIO, and Kristin was responsible for presenting weekly progress to Eva.

A prototype methodology was used to develop Ankh. We started out by making a simple prototype of Ankh with reduced functionality and then gradually added additional menu options and more advanced functionality.

We placed a lot of emphasis on testing early and often. We also used Ankh itself to develop Ankh, which gave us a deeper insight in the strengths and weaknesses of the product, and made it easier to resolve bugs.

We aimed at avoiding duplication of code lines. If more than three code lines were used more than once, usually the common functionality was factored out into a new method. We attempted as far as possible to reuse already existing methods.

The technical challenges the group has been exposed to have mainly been solved internally within the group or through mailing lists, news groups and books. We have aimed to avoid inflicting Ankh technical issues on our supervisors.

We made the project specification and work schedule ourselves. During the project period we experienced that it was difficult to make realistic plans and follow the time schedule, but in the end we found that we did not deviate considerably from the original project specification, taking into account the challenges we have gone through.

The tools used are described in detail in the appendix and subchapter 2.4 Learning phase and tools used.

Ankh became a part of the well-established open source community at <http://ankhsvn.tigris.org> around Easter. We communicated with the users of Ankh through the ankhsvn mailing list. The interest shown has been encouraging, if not overwhelming.

2.2 Organising an optimal process environment

We spent some effort at developing an optimal process environment. As we planned to work separately for periods of time we needed a system where we could follow the project progress at anytime from anywhere. Since the project was about Subversion, it followed naturally that we would use Subversion for the project itself. To be able to at any time get the last update of the project we reviewed information from commit mails, build mails, the diary and had code reviews in addition to SVN revision control. Commit mails, build mails and the diary are described in the sections below.

2.2.1 Commit mail

Every time a project member performed a subversion commit, an e-mail was sent to all group members. The e-mail contained a log message written to the specific revision and an “svn diff” that contained information about which files and directories that had been modified, added or deleted and which sentences that had been added or removed in each file (not binary files).

```
[COMMIT] Commit by Per to "Final year project" on 2003-04-10 00:29:54 +0200 (Thu, 10 Apr 2003) - ...

Fra:      svn-commit@ankhsvn.sytes.net
Til:      svn-commit@ankhsvn.tigris.org
Kopi:
Emne:     [COMMIT] Commit by Per to "Final year project" on 2003-04-10 00:29:54 +0200 (Thu, 10 Apr 2003)

Commit by Per to "Final year project" on 2003-04-10 00:29:54 +0200 (Thu, 10 Apr 2003).
HEAD revision is now 360

Changed:
U   trunk/src/Ankh/Commands/RevertItemCommand.cs

Log message:

Forgot to make RevertItemCommand inherit from CommandBase.

M   /src/Ankh/Commands/RevertItemCommand.cs - fixed inheritance

Diff:

Modified: trunk/src/Ankh/Commands/RevertItemCommand.cs
=====
--- trunk/src/Ankh/Commands/RevertItemCommand.cs      (original)
+++ trunk/src/Ankh/Commands/RevertItemCommand.cs      2003-04-10 00:29:56.000000000 +0200
@@ -4,6 +4,7 @@
 using NSvn.Core;
 using NSvn.Common;
 using EnvDTE;
+using Ankh.UI;

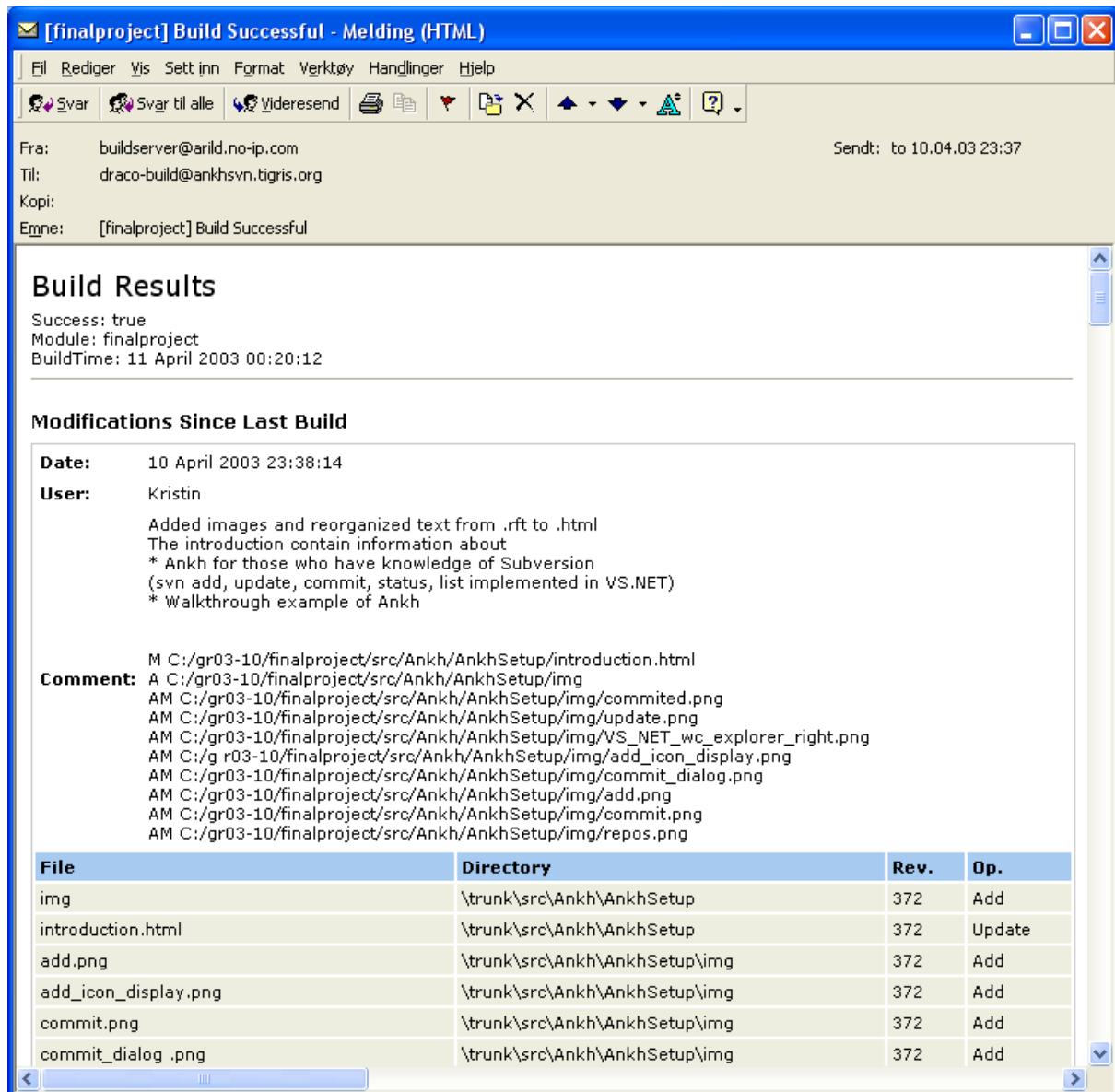
 namespace Ankh.Commands
 {
@@ -16,11 +17,8 @@
     VSNetControl( "Project", Position = 4 ),
     VSNetControl( "Folder", Position = 4 ),
     VSNetControl( "Solution", Position = 4 )]
-
-   public class RevertItem
-   {
-       internal class RevertItemCommand : CommandBase
-       {

```

Using revision control and receiving commit e-mails made it possible to get an overview of what work was performed at any time. If one group member had a problem that was difficult to solve another group member could easily get information about the problem and provide support to the other member.

2.2.2 Build mail

At regular intervals, a NAnt build outside VS.NET was performed. As part of this build all unit-tests were run to ensure the sanity and quality of the system at any given moment. An e-mail was sent to each group member with information about whether the build and tests ran successfully. To accomplish this, we used the Draco.NET continuous integration build server product. Draco supported some revision control systems (CVS, VSS, PVCS) but not Subversion. We therefore contributed code to the Draco project to support Subversion.



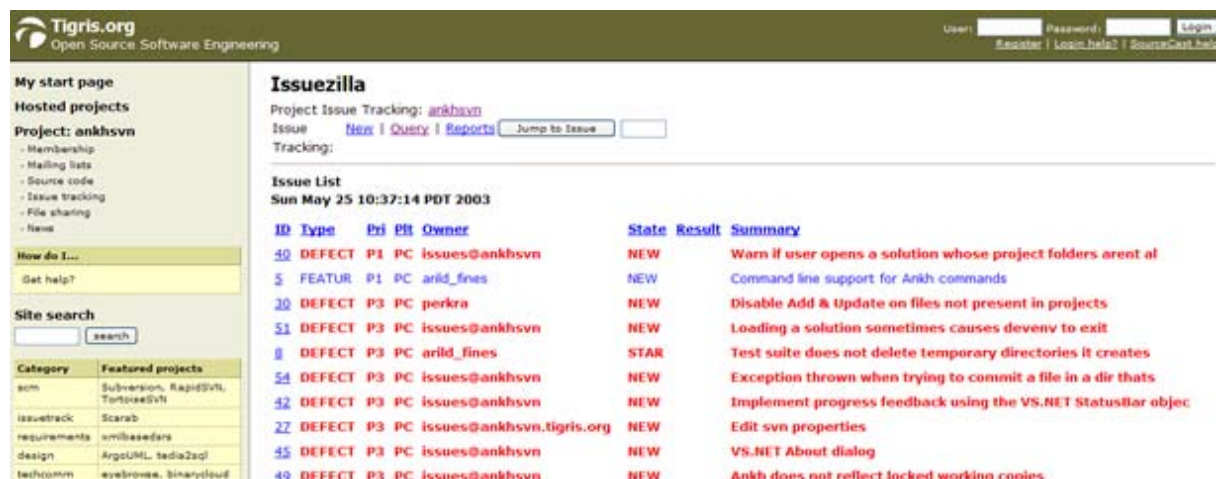
2.2.3 Diary and Internet sites

The Internet site was adjusted to give focus on Ankh as a product, and not merely as a final year project at HIO. The diary had both a public and a private part. Both supervisors had write access to the diary. The diary provided information of the Ankh project; a news part where important milestones were published and a part where we wrote information on a day-to-day basis.

In the last part of the project period Ankh became a part of the well-established open source community at <http://ankhsvn.tigris.org>. Ankh obtained a number of tools from this site, among them the opportunity to set up custom mailing lists. We did get a number of observers showing interest in the Ankh project, and we communicated with them through the mailing lists. The interest shown has been encouraging.

2.2.4 Issue-tracker

Another tool provided to us was an issue tracker. We started to use this tool close to the Alpha release. The issue tracker keeps track of current weaknesses and possible future improvements to Ankh. The issue list shows type of feature, priority, status, result and summary of the issues. This list helped us to remember things we planned to implement in later versions and gave us the status on issues to be solved in near future.



The screenshot shows the Tigris.org Issuezilla interface. The top navigation bar includes the Tigris.org logo, the text "Open Source Software Engineering", and user login fields (User, Password, Login) along with links for Register, Login help, and SourceCode help.

On the left sidebar, under "My start page", there are links for Hosted projects, Project: ankhsvn, Membership, Mailing lists, Source code, Issue tracking, File sharing, and News. Below this is a "How do I..." section with a "Get help?" link. Further down is a "Site search" section with a search box and a "search" button. At the bottom of the sidebar is a "Category" section with a table of featured projects.

The main content area is titled "Issuezilla" and shows "Project Issue Tracking: ankhsvn". It includes links for "New", "Query", "Reports", and a "Jump to Issue" button. Below this is an "Issue List" section with a timestamp "Sun May 25 10:37:14 PDT 2003".

ID	Type	Pri	PR	Owner	State	Result	Summary
40	DEFECT	P1	PC	Issues@ankhsvn	NEW		Warn if user opens a solution whose project folders arent at
5	FEATUR	P1	PC	ankh_fines	NEW		Command line support for Ankh commands
20	DEFECT	P3	PC	perkra	NEW		Disable Add & Update on files not present in projects
51	DEFECT	P3	PC	Issues@ankhsvn	NEW		Loading a solution sometimes causes devenv to exit
8	DEFECT	P3	PC	ankh_fines	STAR		Test suite does not delete temporary directories it creates
54	DEFECT	P3	PC	Issues@ankhsvn	NEW		Exception thrown when trying to commit a file in a dir thats
42	DEFECT	P3	PC	Issues@ankhsvn	NEW		Implement progress feedback using the VS.NET StatusBar objec
27	DEFECT	P3	PC	Issues@ankhsvn.tigris.org	NEW		Edit svn properties
45	DEFECT	P3	PC	Issues@ankhsvn	NEW		VS.NET About dialog
49	DEFECT	P3	PC	Issues@ankhsvn	NEW		Ankh does not reflect locked working copies

2.2.5 Relations to the employer CollabNet

CollabNet did not provide any software or computer equipment. We communicated with CollabNet via e-mail. We did not arrange any meetings, as CollabNet is located in the USA. Throughout the development phase we aimed to inform CollabNet about our progress. We sent e-mail when we had reached important milestones and frequently used our diary. CollabNet helped us in the process to be a part of the well-established open source community at <http://ankhsvn.tigris.org>.

Development process

This chapter covers the development process in the Ankh project. We passed through phases of learning, design, implementation/development, and testing. This chapter describe general challenges we faced during in the development, which tools are used for what purpose, what we did within each phase, alternative approaches, and arguments for the selected GUI.

2.3 Pilot project

In this phase we wrote the progress report, the project proposal, contacted CollabNet, installed software on the school PC, coded the diary, made the work plan and work schedule.

2.4 Learning phase and tools used

During our project we continuously used Subversion for revision control. Revision control systems are not a part of any lectures at HIO so we had to learn how to use Subversion and understand parts of the Subversion code (whose API is coded in C) to be able to create the Ankh NSvn wrapper layer. To get a better understanding of the Client API in Subversion we frequently referred to the SVN API documentation that was provided through a tool called Doxygen. The Doxygen tool made it possible to extract comments from the SVN API source code and read the documentation in a web browser.

To become familiar with the VS.NET environment we explored features and ran macros and samples available through the help utility. During the development phase we frequently used the integrated help utility in VS.NET.

The code was mainly written in C# except for a Managed C++ layer in the NSvn wrapper. To get familiar with the C# programming language we started to look at tutorials in C# and ran samples advised by the VS.NET help utility.

To gather information about VS.NET addins we searched the Internet, tested some available free addins, read articles about the VS.NET addin architecture and used the help utility in VS.NET.

To get an overview of NUnit and NAnt we looked at tutorials.

The CASE tool Enterprise Architect was used to develop use case and class diagrams as one of the group members was familiar with this tool and could easily transfer his knowledge to the rest of the group.

The diary was programmed in Python as one of the group members was familiar with the language and we were not sure whether a Tomcat server would be available for two alternative diaries made last year in the Distributed Information Systems course. Most of the diary at our Internet site was programmed during the Christmas holiday last year, but the site was maintained and adjusted throughout the whole project period.

A more detailed description of tools used is described in the product report and in the appendix in this report.

2.5 Design phase

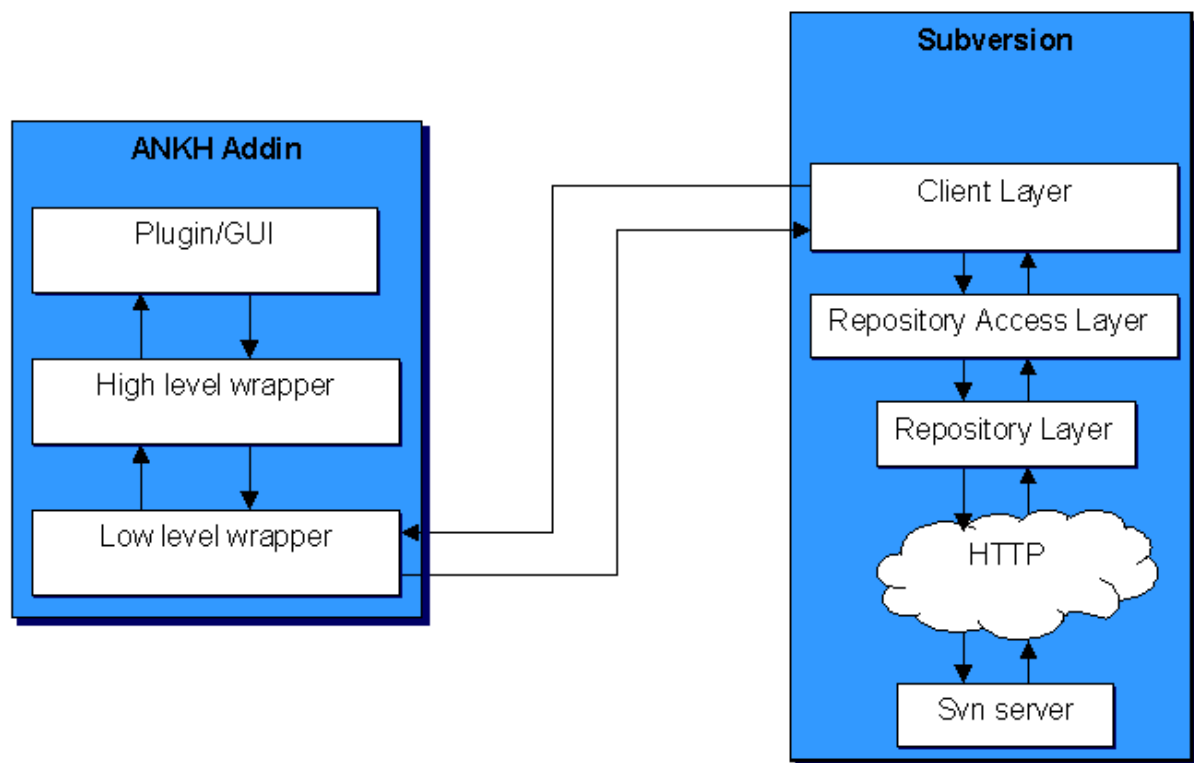
In this phase we agreed on system design, analysed the potential users of our program, tried to figure out what Ankh could do for the user in a mental model and discussed the navigation model and menus to be used.

A conceptual model of Ankh from the developer's point of view was made where we analysed each menu-option in Subversion and how it could be implemented in VS.NET. These use case diagrams and schemes are enclosed in the product documentation.

Common programming guidelines were composed before we started the coding to get an overall consistent look of the code. These guidelines are described in the product documentation.

2.5.1 System design

The system is divided into two main layers, a GUI/addin layer and a .NET wrapper layer. The addin/GUI layer contains the addin itself and code for graphical user interface written in the C# language. This layer is called Ankh. The wrapper is divided into a lower level part for the SVN's client API developed in Managed C++ and a more object-oriented high level part written in C#. The .NET wrapper is called NSvn.



2.5.2 Analysis of the users of our program:

The users of Ankh:

- Have programming experience.
- Use or have knowledge about VS.NET.
- Have knowledge about revision control systems.
- Are fastidious concerning design and user-friendliness.
- Are usually male.
- Are often within the age of 20-45.
- Are probably members of a project group
- Choose to use our addin because a version control system is a useful tool in development of a potential software product
- Might have prior knowledge of Subversion or CVS (the successor to Subversion)

Ankh is made for a programming environment. We concluded that ordinary people without programming experience would not represent Ankh users.

2.5.3 Mental model

A mental model from the users' point of view was made to describe the functionality of the wrapper and what Ankh could do for them, if we were able to implement all commands from Subversion. Two actors were identified; the programmer that wants to use revision control and the repository that could be compared to a server (with some additional features). Relations between the objects and the actors are shown with lines. An item means a file, folder, project or solution.

Mental model of Ankh



2.5.4 Navigation model and menus

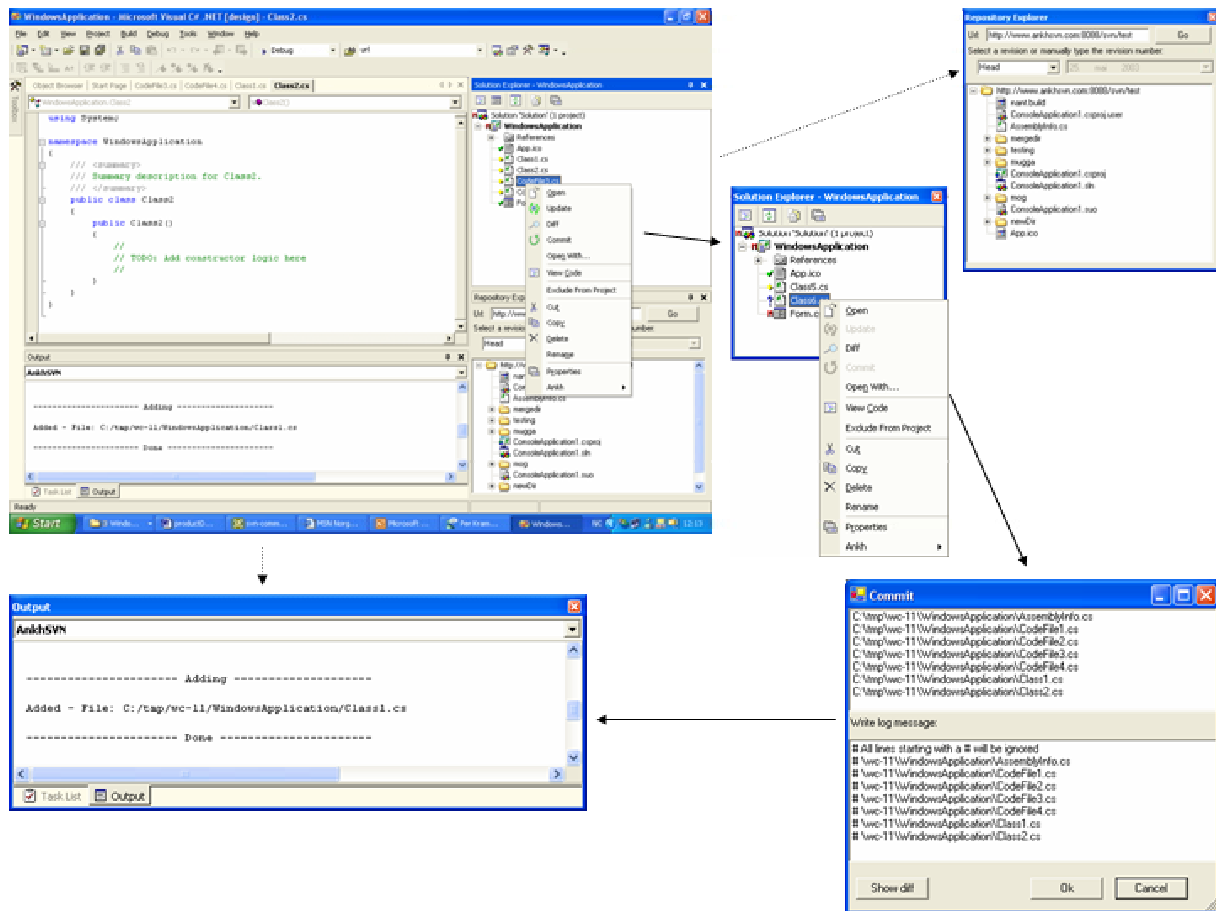
Our goal was to implement Subversion into VS.NET as smoothly as possible. The VS.NET GUI was rather complex with lots of windows, menus and icons. Ankh was supposed to be a supplementary tool to VS.NET. Making a fancy design in bright colours would probably just be distracting to the programmer who is supposed to be concentrating on coding, so we decided to use the existing GUI as much as possible. We added a new pane to the output view, added icons to the Solution Explorer, added a new tool window for the Repository Explorer and added new menu options to the existing context menus for files, directories, projects and solutions.

The name Ankh, which is the name of our addin, was chosen because it is Egyptian and can be visualised as a simple icon that easily can be used as a logo in the marketing of Ankh. It was important to select an icon that could not be confused with another well-known software product and that had a neutral meaning. As projects on the tigris website tended to use Egyptian words we decided to follow their trend.

The menu choices are as functional and obvious as possible. For example the command "svn commit" is named "commit" in the context menu. There might be some programmers that are already familiar with SVN. "Publish" could have been a more understandable term than "commit", but since we decided to take into account users already familiar with SVN we have avoided new terms.

Our navigation model is described in the figure below. If the programmer right clicks on a file, folder, project or solution (item) a menu will pop up. If the command requires further input, a dialog box will be displayed. The result of the action initiated in the dialog box or from the item menu will be displayed in the AnkhSVN pane in the

Output view. There is also a new tool window containing a file tree view of the repository.



2.6 Implementation phase

Before we started writing any code we agreed on common programming guidelines to get an overall consistent look of the code. The programming guidelines are described in detail in the product documentation.

During the development phase we continuously tried to assure that we worked within the frames of the project specification. Use case diagrams and scenarios were used as basis for the coding.

We started the writing the wrapper layer in parallel with the development of the GUI (dialogs and user controls). The project specification was important in the choice of the selected GUI. It was not possible to start the VS.NET integration before the wrapper layer was completed.

When we started to work with the GUI layer we arranged a paper prototype test of the GUI to ensure that our navigation model was logical and intuitive. This test is described in the test report. Based on feedback from the paper prototype testing, we realised that some kind of integrated help functionality was required so Ankh Help was made.

The following subchapters describe the challenges we were exposed to during the coding phase, alternative approaches, improvements of the GUI compared to today's situation and reasons for the selected GUI.

2.6.1 General project challenges

New versions of Subversion were released about every third week as Subversion is still under development. Consequently we had to adjust the NSvn layer to new versions a number of times.

This was a time consuming process. We started with SVN version 0.16 and ended with version 0.22.2.

As the project progressed Subversion unexpectedly changed their Client API to include more flexible authentication architecture. This introduced a new client context structure that included client specific callbacks and batons, which served as a cache for configuration options and other various data. These changes caused a temporary setback for us.

In the middle March Microsoft announced a potentially serious bug in VS.NET that only affects C++ developers. The bug could occur while using Managed C++ Class Libraries that usually contain a mixture of native code and intermediate language (IL). The build product of a "Managed C++ application" is an assembly of IL with an .exe extension. As we are using mixed DLLs this bug can potentially affect our addin. A possible suggestion of a "workaround" solution for this bug was to set the build output in the Managed C++ Class Library project to have an IL-only file. As we had limited time available and did not experience a big impact of this bug we have not implemented the "workaround".

The most difficult part of the Ankh project was the integration into VS.NET because this area was poorly documented and few relevant examples were available. The examples available were mainly macros coded in Visual Basic. To gather information in this phase we used news channels, mailing lists, books and searched the Internet. As the Ankh project is open source we hope to transfer our knowledge of the VS .NET integration coded in C# to others. The Ankh project represents a complete example of how to hook up to menus, make explorer panes, output panes and how to make dialog boxes and user controls coded in C#.

2.6.2 NSvn wrapper layer

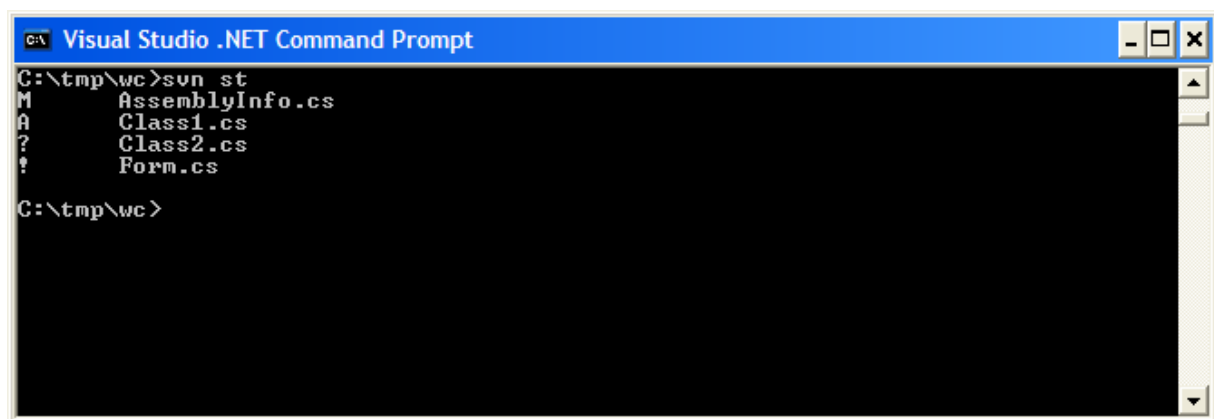
We named the VS.NET wrapper NSvn. The choice of the name selected was based on the short name of Subversion, SVN, combined with the letter N symbolising .NET. This follows the general pattern from other open source .NET efforts such as NUnit, NAnt and NDoc. We divided the wrapper in two levels. A low-level part was coded in Managed C++ and a more object-oriented higher-level part was coded in C#. NSvn contains the functionality of all the "svn_client" functions described in the SVN Client API. We wrote unit-tests to verify the translation for all "svn" commands. The .NET wrapper layer is available as a standalone component so that it can easily be used in other applications in the future.

We primarily intended to write most of the system in the C# programming language, but discovered that writing the .NET wrapper layer in Managed C++(C++ extensions for the .NET environment) was a better choice since Managed C++ offered a better integration with C-based APIs such as that of Subversion. Another potential alternative was to use a tool called SWIG (Simplified Wrapper and Interface Generator) - a tool that generates wrappers for C and C++-based APIs automatically. The C# module had recently been added to SWIG and was still in a state of development. We researched these alternatives and sought advice from people with knowledge in the area on the Subversions mailing list. The list response was mainly supportive of the choice we ended up making.

2.6.3 Integration of status icons in Solution Explorer

This subsection describes the integration of the SVN command status into VS.NET's Solution Explorer to give an indication of the GUI improvements and challenges of the integration.

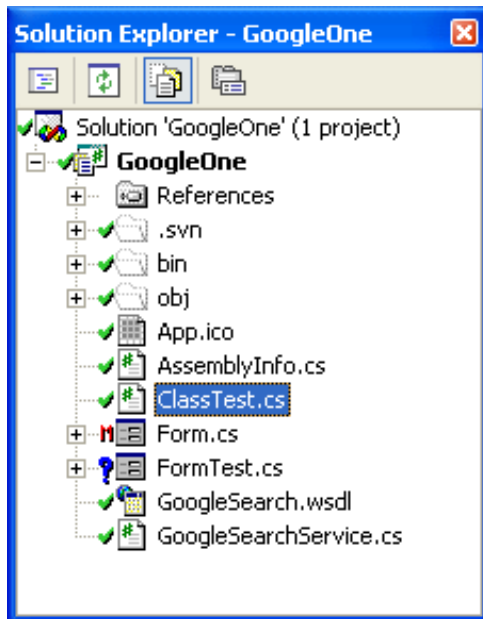
The picture below shows an example of how SVN works today in a command prompt.



```
C:\ Visual Studio .NET Command Prompt
C:\tmp\wc>svn st
M      AssemblyInfo.cs
A      Class1.cs
?      Class2.cs
!      Form.cs
C:\tmp\wc>
```








The “svn status” command prints the status of files and directories in a working copy. A working copy is a collection of files and directories under revision control. The symbols in front of each file indicate the status of each file or directory.

The Subversion command “svn status” was integrated into VS.NET in Solution Explorer. Solution Explorer display files, directories, projects and solutions (item) with updated Ankh status icons in front of each item. The status icon visible in the Solution Explorer makes it easier to see which files have changed status while working.







By using icons we hope to help the user to reduce his cognitive strain in the long run.

The meaning of the different status icons are listed below:

-  Unversioned item: Item not under revision control.
-  Versioned item: Item under revision control.
-  Modified and versioned item.
-  Added and versioned item.
-  Conflicted, versioned item: If two users modify the same file at the same line a conflict will occur if the file is under revision control. The conflict occurs because the repository simply doesn't know whether both or only one of the lines should be added.
-  Deleted versioned item.
-  Locked and versioned item: A locked item is an item that is not accessible. For example during a commit an item is not accessible for additional changes – it is locked.

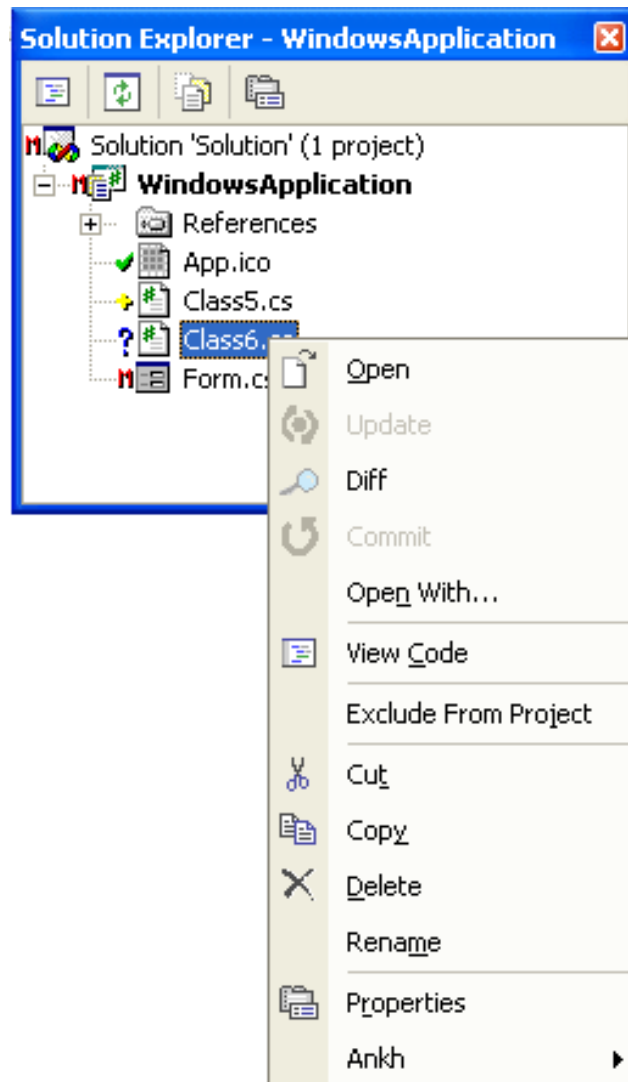
The red colour of the icons is used to indicate danger or attract attention. If an item is modified it could mean that some text is scheduled to be removed in a potential commit. If an item is conflicted or locked, a warning message will be displayed. It is important to solve a potential conflict and cleanup locked items so the colour red is used to attract attention to these items.

The green colour on the versioned item icon () signals that everything is ok. If a commit succeeds the () icon will be displayed in front of the committed item. The yellow colour on the added, versioned item icon () signals that the item is ready (scheduled) for commit. Blue colour on the unversioned item icon () signals a new category of items which is not versioned.

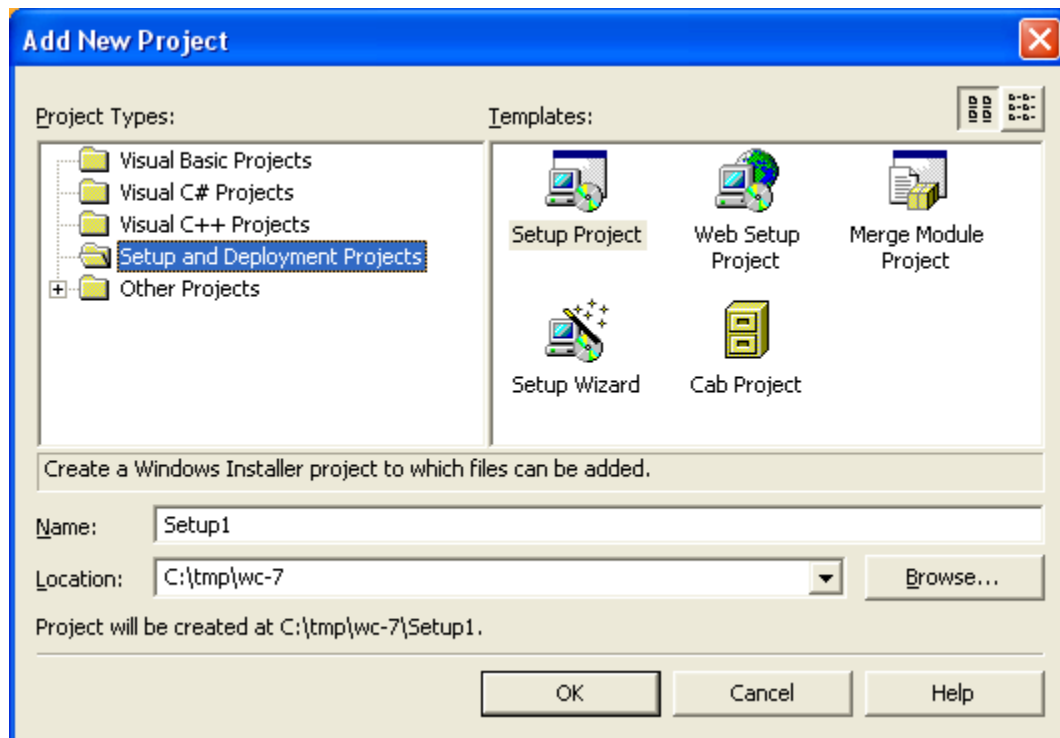
Ankh updates status icons automatically after the status has changed. A recursive refresh command is also available in the item menu for the solution, individual projects and folders.

2.6.4 Ankh integration of menu options into VS .NET

This subsection describes the Ankh implementation of the SVN commands add, update, remove/delete and revert into VS. Below is the context menu for a file.

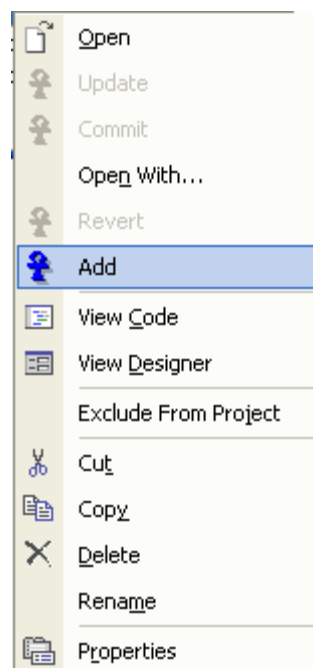


In VS.NET menus are implemented not only for files and directories, but also for projects and solutions. Below is the Add New Project dialog with a list of some of the different project types in VS.NET.




Whenever VS.NET deletes an item from a solution in a working copy, Ankh must catch the event and exclude or delete the item automatically from the working copy. The event fired by VS.NET is different for C++ items compared to items in the other programming languages, something that complicated matters.

Initially we intended to use the Ankh icon in front of all menu options in the item menu mainly to separate Ankh options from VS.NET options in the same menu. The first versions of Ankh therefore had an item menu as shown below.

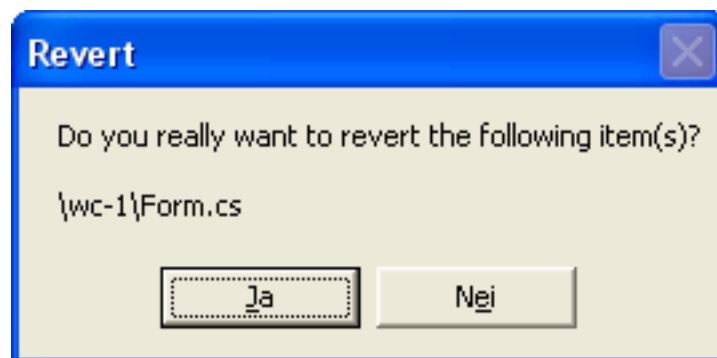


We reconsidered this approach. As we hope to reach a new audience that is not necessarily as familiar with the terms used we hope these icons can help the user to learn the new terms. The final Ankh icons used are meant to help the user to remember the meaning of the menu options and thereby reduce the users cognitive load. We believe it is easier to remember a picture than to remember a new technical term.

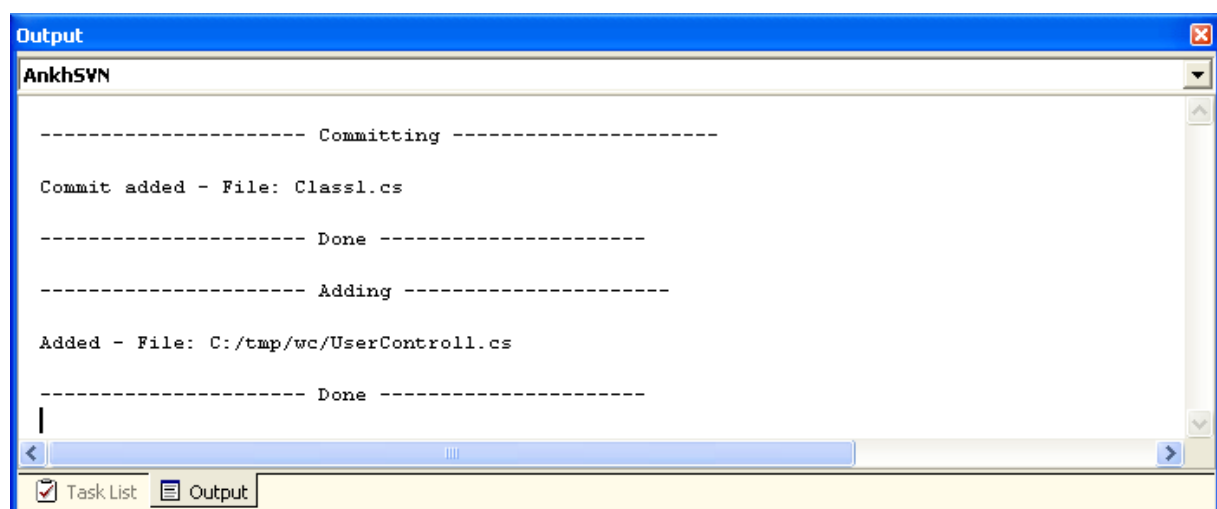
A magnifying glass  was used in front of the Diff option to help the user remember that the Diff option displays details about what text has been removed and/or added since last local commit or update.

Revert can be confused with “undo” as it reverts changes performed in the working copy and brings the working copy back to the pristine state. The pristine state is the last updated working copy or working copy from last local commit.
. The icon displayed in front of the revert option in the item menu was created to look like undo for this reason.

If the revert option is selected in the item menu for a specific item, the Revert dialog is displayed. The Revert dialog below was integrated by user request after the Alpha release. Details from this release are given in the test report.

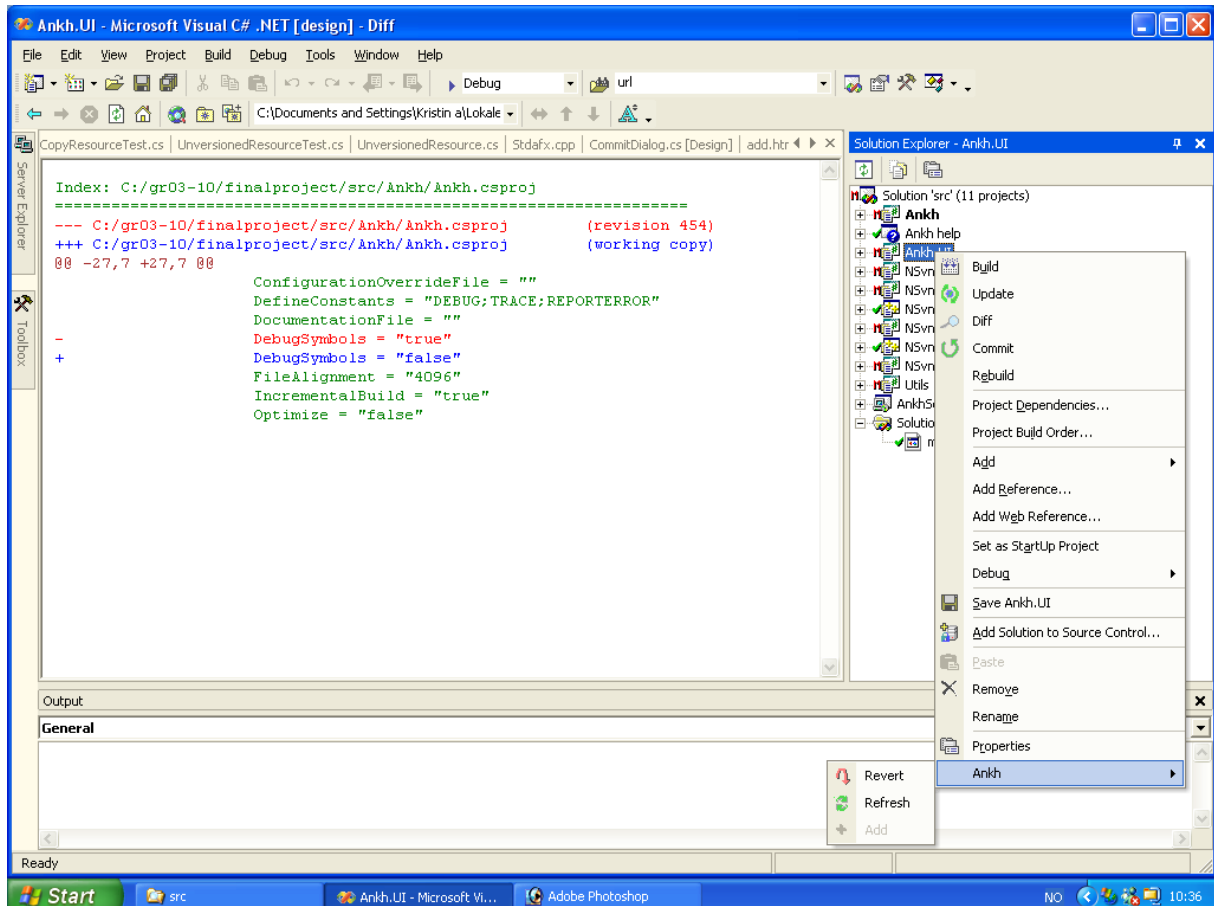


We used an output pane in VS.NET to display feedback from item menu options.



2.6.5 Integration of Diff in VS.NET

This subsection describes the Ankh integration of the SVN command diff into VS.NET to give an indication of the GUI improvement. The Diff command is available from the context menu in Solution Explorer or from within the Commit dialog. A diff displayed in the VS.NET integrated web browser is shown below.

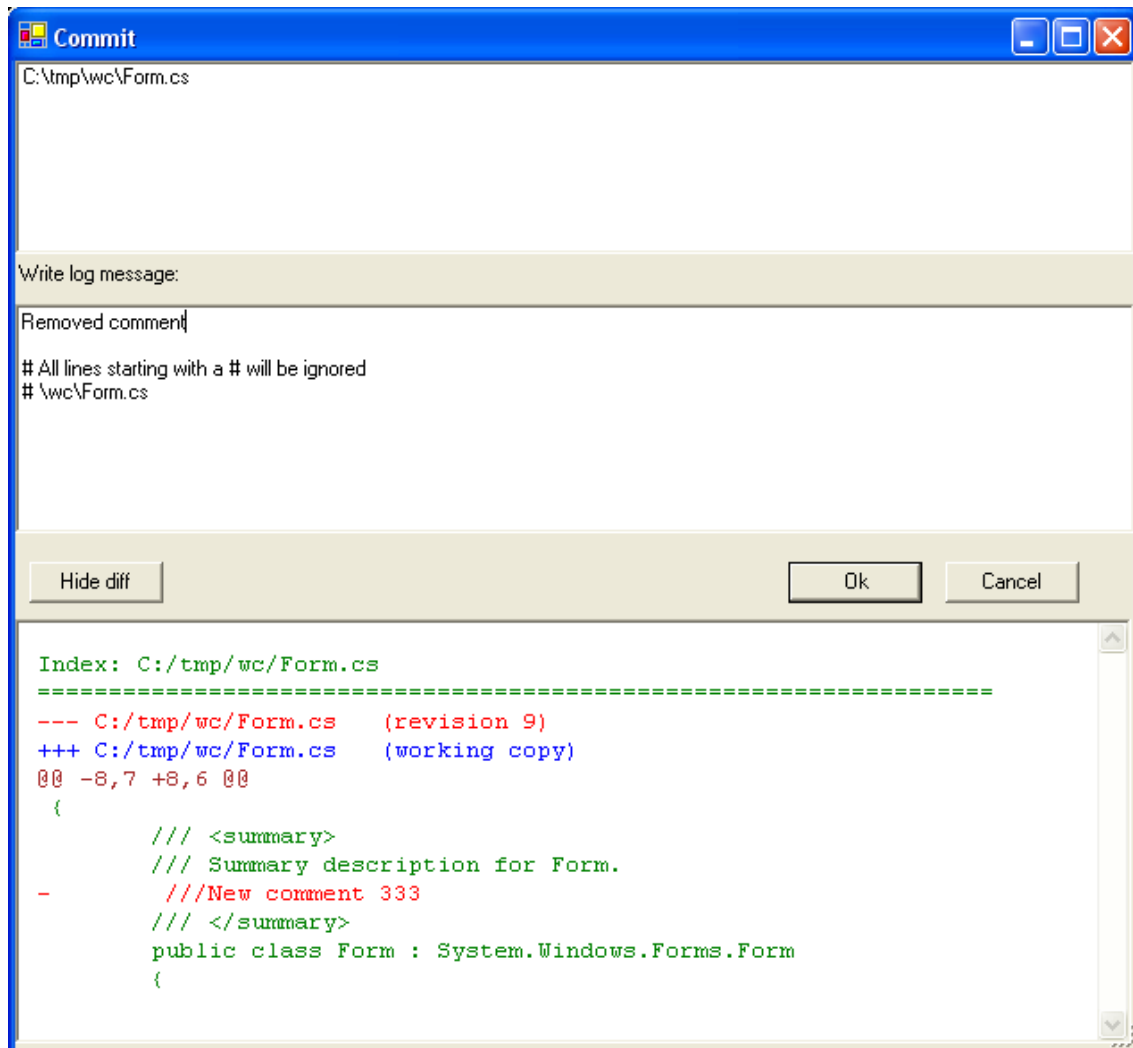


The Diff command lets the developer examine the changes made in the working copy since the last update or commit.

We used colours to show the division between the different categories of text. The colour red was used to indicate text that was removed. Green was used to indicate unchanged text. Blue was used to indicate text added since the last commit or update.

2.6.6 Integration of Commit in VS.NET

This subsection describes the integration of the SVN command commit into VS.NET

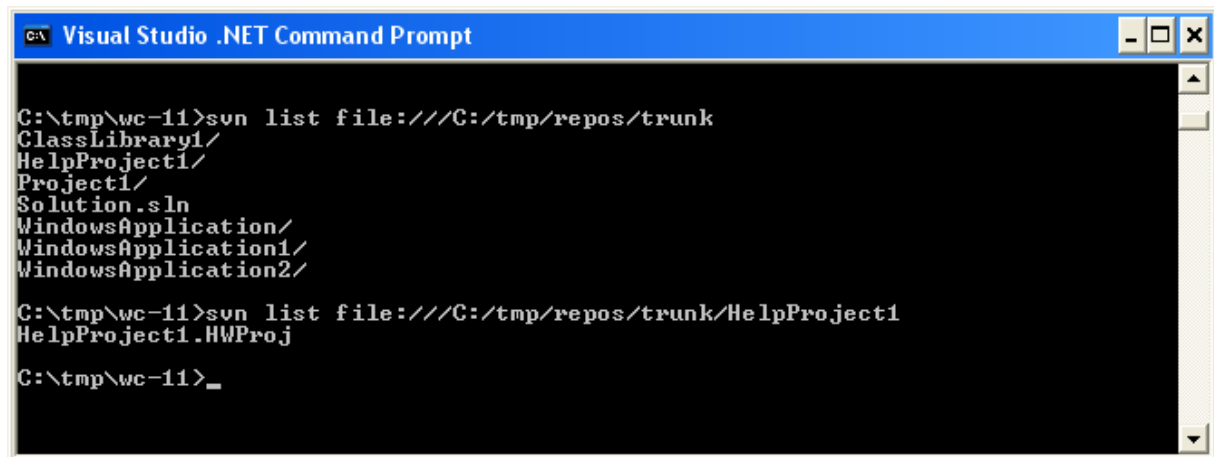


We discovered that it was often useful to visually see the changes we made while writing a log message. We often did several changes before a commit and it was difficult to remember all of them. The Get diff button in the Commit dialog displays all changes made since the last local commit or update.

2.6.7 Integration of file tree view of repository in VS.NET

This subsection describes the integration of the SVN command list into Ankh's Repository.

The picture below shows an example of how the SVN command list works today in a command prompt.

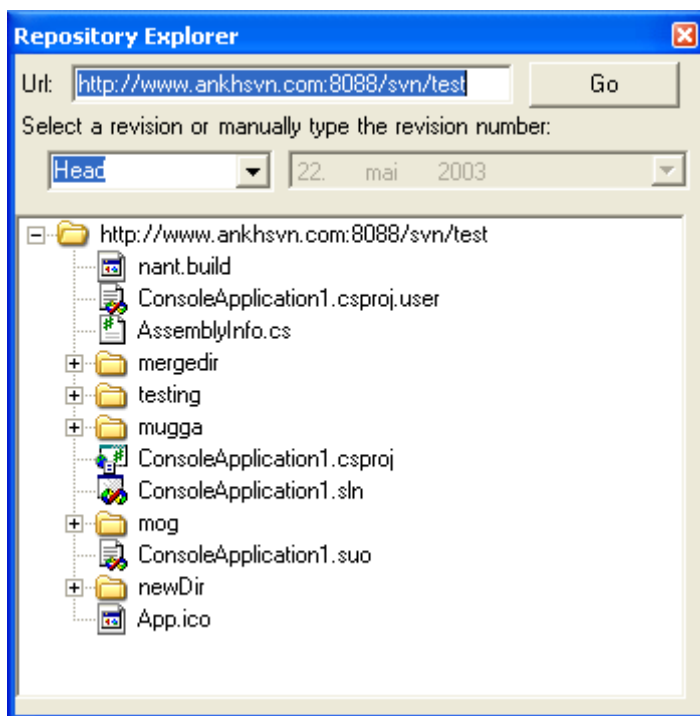


```
C:\tmp\wc-11>svn list file:///C:/tmp/repos/trunk
ClassLibrary1/
HelpProject1/
Project1/
Solution.sln
WindowsApplication/
WindowsApplication1/
WindowsApplication2/

C:\tmp\wc-11>svn list file:///C:/tmp/repos/trunk/HelpProject1
HelpProject1.HWPProj

C:\tmp\wc-11>_
```

The “svn list” command prints a list of the content within the directory in the repository defined based on the URL. If the repository contains a large number of files and directories it is required that the list command be done a number of times to get a complete overview of the repository by using the prompt. We implemented the list functionality as a custom tool window called the Repository Explorer.

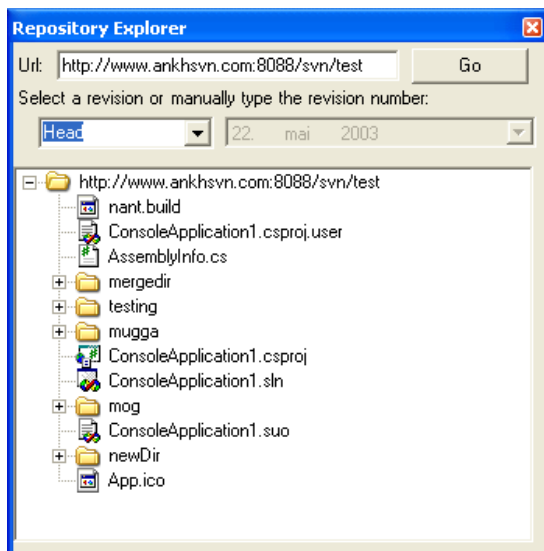
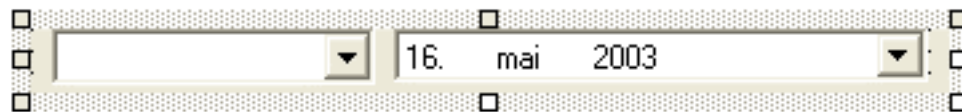


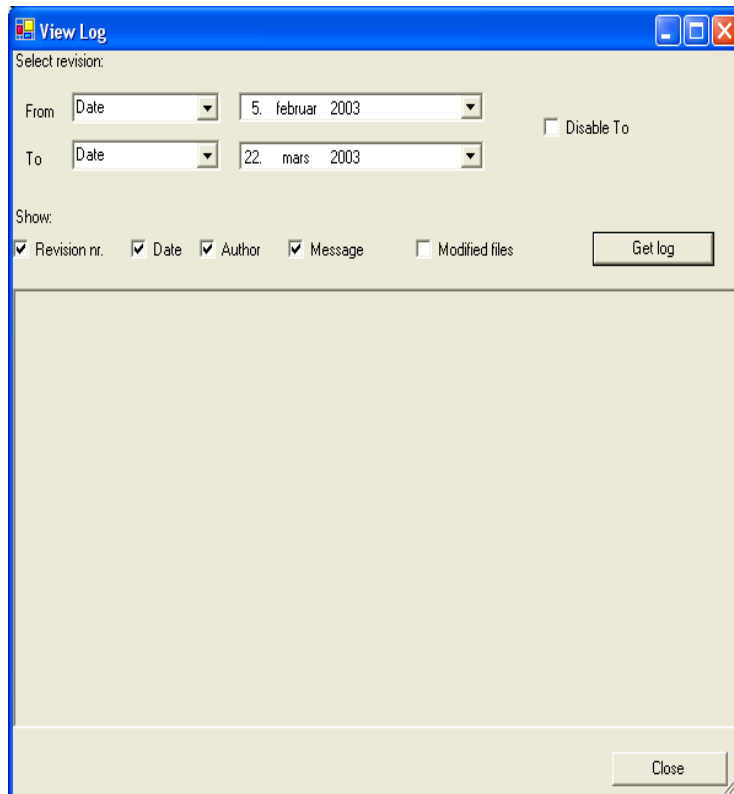
We used the DatePicker component to allow the user to pick a revision corresponding to a specific date.



2.6.8 Examples of reuse of user controls in the GUI layer

One of our goals was to avoid duplication of code. An example of creative reuse of code is shown in the GUI layer. In the Ankh.UI namespace we divided dialogs into smaller pieces by using User Controls. The RevisionPicker user control shown below was used both in Repository Explorer and in the View Log dialog. Unfortunately, we did not manage to integrate all dialogs made in the Ankh.UI namespace, but this namespace represents a complete collection of the dialogs required for the Ankh integration.





Another example of avoiding duplication of code in the GUI layer was with the user controls used in the Edit Properties dialog shown in the subchapter “Avoiding duplication of code lines” in GUI in the appendix.

2.6.9 Ankh Help integration

Keeping in mind the potential new user group, we made it a priority to develop an integrated help system for Ankh. The Ankh Help was meant to be a user guide integrated in VS.NET available from the main menu. The Ankh Help integration covers information concerning basic operations and how to use Ankh.

A method proposed by VS.NET to customise the Help Registration merge module was used for deployment. This method was poorly documented and few examples were available. We asked for advice on mailing lists and news channels, but this did not help us to solve the deployment part of this problem. An alternative approach to deploy the project was to use a tool called H2Reg, but as this tool costs money we decided not to use it. We downloaded the trial version and had a look at the tool. A more detailed description of the problem is contained within in the product documentation.

2.7 Testing phase

Ankh was unit-tested with NUnit and built outside VS.NET with NAnt continuously during the development phase. We tested Ankh functionally by using Ankh while developing Ankh. We released the alpha version to the intended audience at

<http://ankhsvn.tigris.org> for testing. Feedback from these tests gave us useful information about what to be improve and ideas for future development. An early paper prototype testing of the GUI confirmed that our navigation model was logical and intuitive and gave us some useful ideas for how to improve the GUI. In the end we tested Ankh functionality with a number of tests described in the test report to give an indication of the robustness of Ankh. See the test report for more details about testing.

2.8 Documentation

We started to make entries in the diary as soon as it was completed. The first contribution to the diary was October 2002 during the pilot project period. We used the diary continuously throughout the project period. The diary was used as basis for writing the process documentation.

We started the documentation when the first prototype of Ankh was made. The process, product and test documentation was written in Word. All group members have proofread all parts of the documentation. The employer did not require a user manual.

2.9 Deviations from project specifications

The project specification was not changed during the development process.

The project specification contained useful guidelines that were used both during the design and implementation phase. The selected GUI and the final code of Ankh are optimised based on the project specifications.

Process, test and product documentation was written in Word instead of XML due to strict time-schedule and unforeseen challenges. We started to write the documentation in XML. XML/XSL-documentation is suited for small documents, but among other things it has no option for spell check. To get table of contents, page numbering that takes into account different images sizes, headers/footers etc. complicated XSL-code would have been required. As the HIO did not require both an optimised paper and web-version of the documentation we prioritised the quality of the content instead. We consider this deviation from the project specification of minor importance.

Common guidelines for how to write log messages in Subversion were not composed. We considered this deviation from the project specification of minor importance to the final Ankh product.

Throughout our project period we have reached our target. We can provide an integration of SVN into the existing development environment VS.NET. Ankh is available for anyone from anywhere through the well established open source community at <http://ankhsvn.tigris.org>. The .NET wrapper part of Ankh is available as a standalone component that covers the complete functionality of the SVN's Client API.

We have developed an integrated help system that was not part of the original specification.

3 Conclusion

This chapter sums up what we obtained and is an evaluation of the work performed. “Evaluation of the product” describes the physical result of the final project. “Evaluation of the process” describes how we worked and what we learned.

3.1 Evaluation of the product

Improvements due to the development of Ankh:

- By creating a GUI, SVN is made available to a large number of new Windows programmers. Today SVN’s audience is mainly composed of Unix users. We hope to reverse this trend.
- Increased productivity as version control is directly accessible in VS.NET.
- We can offer the Windows users a FREE open source version control tool integrated into VS.NET
- VS .NET integration coded in C# is poorly documented. The Ankh project represent a complete example of how to hook up to menus, make explorer panes, output panes and how to make dialog boxes and user controls coded in C#. Since Ankh is open source code we hope to transfer our knowledge to others.

Ankh covers the functionality most frequently used in daily sessions with Subversion.

The .NET wrapper layer is available as a standalone component so that it can easily be used in other applications in the future. The wrapper covers a complete translation of the Subversions Client API coded in C to Managed C++ and C# code.

Ankh was continuously unit-tested with NUnit and built outside VS.NET with NAnt during the development phase. The functionality of Ankh was tested by using Ankh while developing Ankh. An early paper prototype testing of the GUI was also performed to check whether our navigation model was logical and intuitive. In the end the functionality of Ankh was tested with a number of tests described in the test report.

The Ankh project was developed using advanced C# idioms and patterns. We have explored a new area of VS .NET that was poorly documented, and hope that our knowledge about VS.NET integration will be transferred to others through the open source community at <http://ankhsvn.tigris.org>.

Ankh is available for anybody, anywhere through the open source community at tigris.org. Feedback from observers on the project has been encouraging. The Ankh project will persist in the open source community at <http://ankhsvn.tigris.org>.

3.2 Evaluation of the process

During the process period we have learned how to use Enterprise Architect, Subversion and Visual Studio .NET. We have programmed in C# and Managed C++, learned how to write unit-tests in NUnit, written XML documentation, learned how to make an addin to VS .NET and we have obtained a better understanding of the SVN Client API. We have learned how an open source community works and we have been a part of the open source community at tigris.org through the Ankh project.

There were a number of challenges in the Ankh project. Throughout the project period we have worked with tools that were still in their infancy, and have suffered from it. A possible serious bug was discovered in VS.NET while Subversion was still under development. Ankh was adjusted and built against the SVN source code for a number of SVN versions.

To optimise the working conditions and process revision control was used. Commit e-mails, build e-mails, error e-mails and the issue tracker made it easier to support and encourage each other throughout the development process.

The technical challenges the group has been exposed to have mainly been solved internally within the group or using mailing lists and news groups. We aimed to avoid involving our supervisors in the resolution of Ankh technical issues.

Thanks to Arild who had the splendid idea of the project, represented an amazing knowledge base in the Ankh project, had an enormous working capacity and was the code guru in the group.

Thanks to Per who had an amazing ability to acquire new knowledge, showed an impressive creativity in technical discussions and has a marvellous ability to feel comfortable in whatever working conditions.

Thanks to Kristin for her always optimistic mind, her focus on the users and interest in the marketing part of Ankh.

4 References

1. "NUnit" <http://nunit.org/>
2. "VS.NET help" integrated in VS.NET
3. "NAnt"
4. "Ant" <http://ant.apache.org/manual/>
5. "Creating Office Managed COM Add-Ins with Visual Studio .NET" by Paul Cornell, Microsoft Corporation, June 6, 2002
6. "EA" at <http://www.sparxsystems.com.au/>
7. "The Mixed DLL Loading Problem Using Visual C++ .NET", March 16, 2003
<http://www.codeguru.com/dll/kmg15.html>
8. "15 SWIG and C#" <http://www.swig.org/Doc1.3/CSharp.html>
9. "Subversion: The Definitive Guide", Draft: Revision 5113, 26 February, 2003,
by Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato,
<http://subversion.tigris.org>
10. "Inside Microsoft Visual Studio .NET" by Brian Johnson, Craigh Skibo, Marc Young
11. Draco.NET <http://draconet.sourceforge.net/>
12. Doxygen <http://www.stack.nl/~dimitri/doxygen/>

5 Appendix

5.1 Dictionary

Addin	A software component that extends the functionality of another product."
Alpha release	Alpha is the first release to the general public where basic functionality exists and a reasonable amount of stability is present. Some features may still be absent and there are probably major defects still in the code
API	Application Program Interface
Base revision	Last committed or updated revision
Beta release	A beta release is significantly more advanced than the alpha release. While still not completely stable because of lingering bugs, all features planned for the final release should be present and reasonably reliable. This is when the more elusive bugs will be become apparent.
COM	Component Object Model (COM), a software architecture that allows the components made by different software vendors to be combined into a variety of applications. COM defines a standard for component interoperability. It is not dependent on any particular programming language, is available on multiple platforms, and is extensible.
Commit	Send changes from your working copy to the repository.
Conflicted file	If two users modify the same file at the same line a conflict will occur if the file is under revision control. The conflict occurs because the repository simply doesn't know whether both or only one of the lines should be added.
CVS	Concurrent Versions System -A free open source version control system, the predecessor to Subversion.
DLL	Dynamic Link Library
Doxygen	Doxygen is a documentation system for C++, C, Java, IDL (Corba, Microsoft, and KDE-DCOP flavours);and to some extent PHP and C#. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual from a set of documented source files.

Draco.NET	Draco.NET is an automated build system. It monitors your source repository, automatically rebuilds your project when changes are detected and e-mails you the build result along with a list of changes since the last build. Draco.NET version 1.4 supports the NAnt build tool. It now also supports Subversion as we contributed code to their repository.
Folder	Directory
GUI	Graphical User Interface
Head revision	Last revision in repository
Item	File, folder, project or solution
IDE	Integrated Development Environment
IIS	Internet Information Server
IL	Intermediate Language, the bytecode to which all languages in the .NET framework compile.
NAnt	A free .NET build tool.
NUnit	A unit testing framework, based on JUnit and part of the xUnit family. .
Repository	The server component of the Subversion version control system; this is where your files get stored.
Repository Explorer	An Ankh tool used to browse a Subversion repository.
Revision control system	In a typical software development environment, many developers will be engaged in work on one code base. Revision control makes it easier for several people to share the same collection of program source code. A revision control system is a centralised system for sharing information where the core is a repository.
Solution Explorer	A window in VS.NET used to manage files in a project/solution
SVN	Short name for Subversion, which is a free open source revision control system.
SWIG	Simplified Wrapper and Interface Generator
UI	User Interface
UML	Unified Modelling Language
Unversioned file	File not under version control
VB	Visual Basic .NET
VC++	Visual C++ .NET
VC#	Visual C# .NET
Versioned file	File under version control
VS .NET	Visual Studio .NET. A programming platform that includes environments for developing C#, C++, Managed C++ and Visual Basic programs.
WC	Short for Working Copy.
Working Copy	Collection of files and directories under revision control.

5.2 Development tools

5.2.1 VS.NET

Visual Studio .NET is a set of development tools for building ASP Web applications, XML Web services, desktop applications, and mobile applications. Visual Basic .NET, Visual C++ .NET, and Visual C# .NET all use the same integrated development environment (IDE), which allows them to share tools and facilitates in the creation of mixed-language solutions. In addition, these languages leverage the functionality of the .NET Framework, which provides access to key technologies that simplify the development of ASP Web applications and XML Web services.

5.2.2 VS.NET addin architecture

Visual Studio NET provides an open development model that allows developers to extend the functionality of the IDE through special components called addins. Since VS.NET itself is implemented in COM, addins also need to be implemented using COM. Fortunately, .NET allows you to expose .NET assemblies as COM components very easily and that is what we did during the development of Ankh. VS.NET also allows you to develop addins using C++ and the Active Template Library (ATL), but this is a lot harder, while it doesn't give any significant benefits over writing it in C#.

5.2.3 NUnit

NUnit is a free open source unit-testing framework for all .Net languages. It was initially ported from JUnit, which is a unit-testing tool for Java(JUnit itself was originally ported from Smalltalk...). We used NUnit 2.0 during the development of Ankh to unit test the classes in the NSvn wrapper layer.

5.2.4 NAnt

NAnt is a free .NET build tool, and serves the same purpose as `make` in traditional environments. Instead of writing shell commands, the configuration files are XML-based, calling out a target tree where various tasks get executed. The various tasks include compiling, linking, copying files, creating zip files and cleaning the source tree from files created by the build process. NAnt is used to generate regular builds of Ankh and NSvn that don't rely on having VS.NET installed on the build computer.

5.2.5 Python

Python is an interpreted, cross platform, object-oriented programming language. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing.

5.2.6 C#

Microsoft C# (pronounced C sharp) is a new programming language designed for building a wide range of enterprise applications that run on the .NET Framework. C# code is compiled as managed code, which means it benefits from the services of the common language runtime. These services include language interoperability, garbage collection, enhanced security, and improved versioning support.

5.2.7 Managed C++

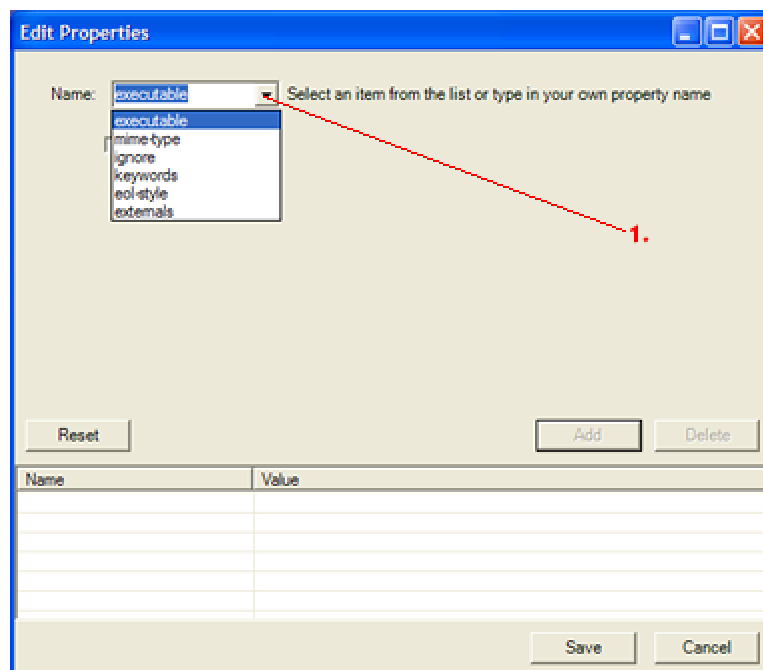
Managed Extensions for C++ are enhancements made to the C++ language, giving C++ full access to .NET functionality. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework. Managed C++ is used to interface with the Subversion C client API and present it to the managed world.

5.2.8 Enterprise Architect

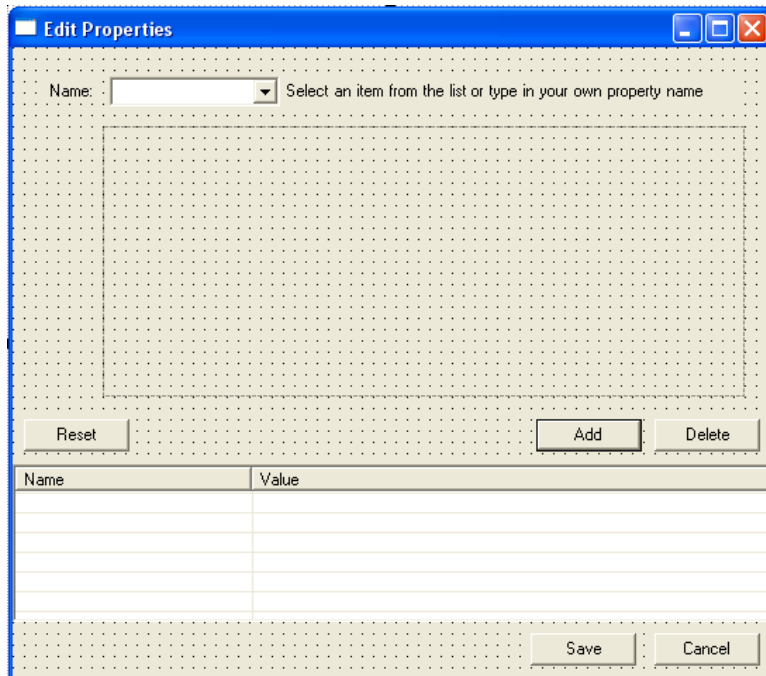
Enterprise Architect is a project oriented UML modelling tool for the Windows platform. This tool was used to generate the use case diagrams and class diagrams.

5.3 Avoiding duplication of code lines in GUI

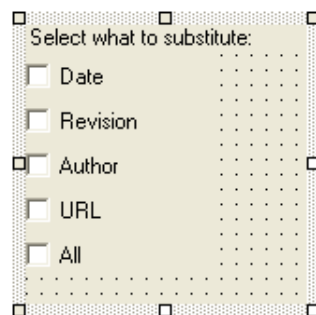
An example of the avoidance of duplication of code lines in the GUI layer is with the user controls used in the Edit Properties dialog.



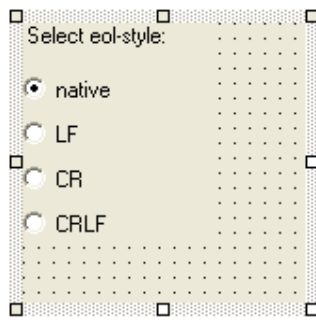
Depending on which menu option (marked as 1 in the above figure) is selected in the Name combobox, different user controls are used. The area where the different user controls are displayed in the Edit Properties dialog is marked with dotted lines in the figure below.



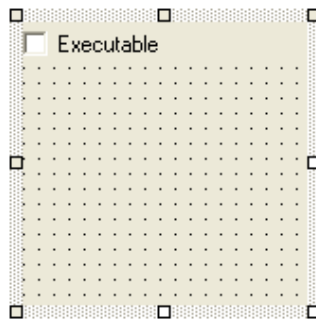
If “Keywords” is selected in the Name combo in the Edit Properties dialog the following user control is displayed in the dotted area of the dialog.



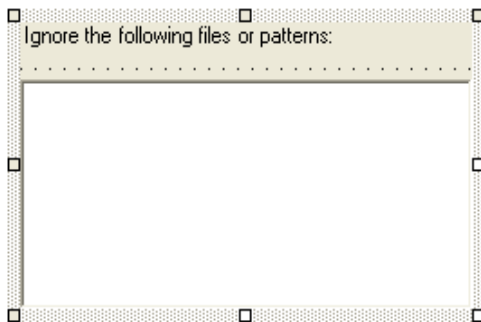
If “eol-style” is selected in the Name combo in Edit Properties dialog the following user control will be displayed in the dotted area.



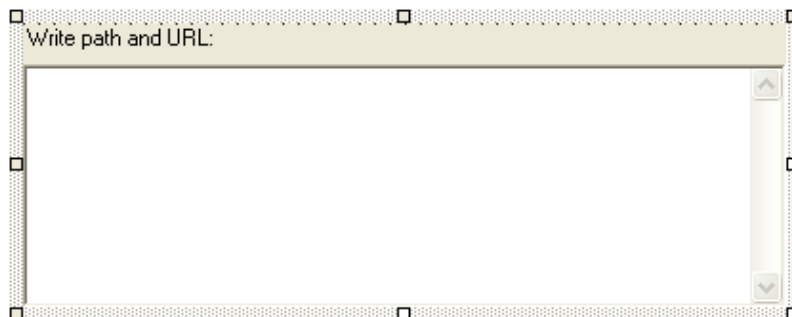
If “executable” is selected in the Name combo in Edit Properties dialog the following user control is displayed:



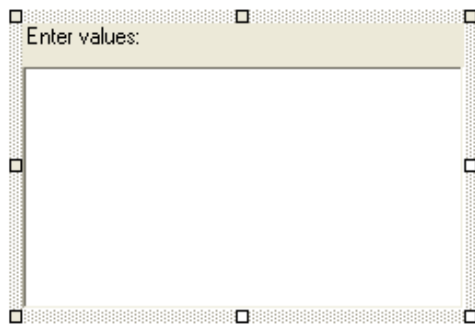
If ignore is selected in the Name combo in Edit Properties dialog the following user control is displayed:



If the user selects externals in the Name combo in Edit Properties dialog, the following user control is displayed in the dotted area.

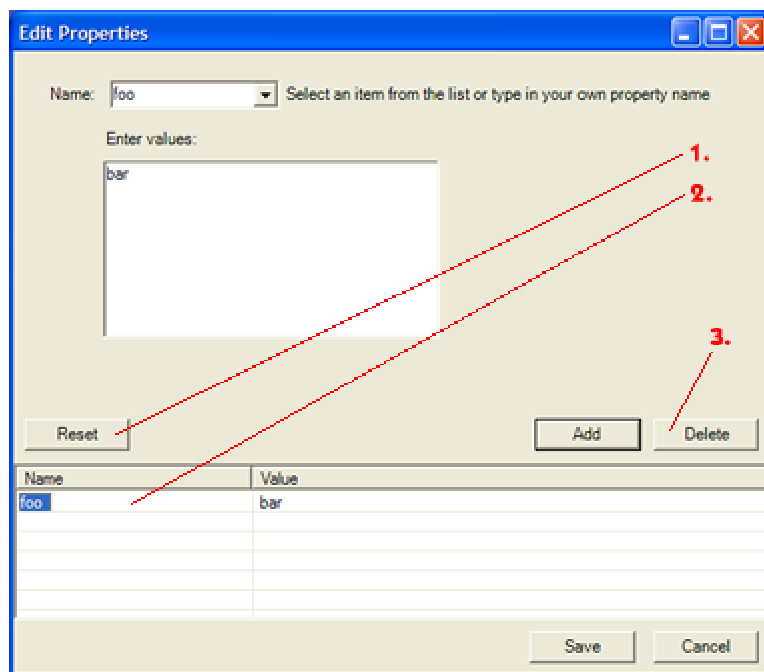


If the user enters a property name of his own choosing, the following control is shown:



A dialog box titled "Enter values:" with a large empty text area for input.

This allows the user to enter custom properties. An example of how the dialog can look if a custom property is defined is shown below.



The "Edit Properties" dialog box shows a list of properties and input fields. Red arrows point to specific elements:

- 1. Points to the "Reset" button.
- 2. Points to the "Add" button.
- 3. Points to the "Delete" button.

The dialog includes a "Name:" dropdown menu with the value "foo" and a text area for "Enter values:" containing "bar". Below the input fields is a table with two columns: "Name" and "Value".

Name	Value
foo	bar

Buttons at the bottom include "Reset", "Add", "Delete", "Save", and "Cancel".