Integration and regular builds

Version	Date	Author	Description
1.4	13.04.2003	Anders Gebala	Added conclusions
1.3	24.03.2003	Anders Gebala	Added experiences from I3
1.2	07.02.2003	Anders Gebala	Added experiences from I2
1.1	02.12.2002	Anders Gebala	Added experiences from I1
1.0	28.10.2002	Anders Gebala	Revised based on mentor feedback
0.5	25.10.2002	Anders Gebala	First Version

Introduction

As the field of software development is maturing the demand for lighter software development processes, has lead a number of so called “agile software development methods” [1]. One of the things these agile methods have in common is that the software development is incremental, consisting of many small releases and rapid cycles. Continuous integration with automated regular builds is some of the things these agile methods emphasize. Extreme programming (XP) considers “integrate often” as a “rule” and Feature-Driven Development considers “regular builds” to be a “best practise” [2], [3]. The idea is to integrate often and solve the small integration problems (or avoid them altogether) while they’re still minor. Regular builds are also a way of ensuring that the rhythm, an important stabilizing and coordinating force in software projects according to Dikel et al, in the project stays the same [4].

Our project

In this project, the developers will integrate and release (commit) their code into the code repository (CVS) whenever they have a suitable opportunity to do so, e.g. when a developer has finished a feature and it has passed the developers’ own testing. By doing this, we ensure that the developers always start from a clean table each time they check out and work with the newest version of the project code. Integration is done as often as possible to minimize eventual problems due to integration and to ensure possible re-use of code. Problems due to incompatibility between modules will be detected early and eliminated before they cause serious problems [5], [6]. A well-planned architecture and use of interfaces between modules help to eliminate these kinds of problems too.

Regular builds (each night) will be used to ensure that the code compiles and to monitor the progress. In conjunction to the builds we will also do some simple tests, so called “smoke tests”, e.g. starting the application. The build tool (Ant) will be configured to send an email after each nightly build to the group members about the success or failure of the compilation and “smoke tests”. Integration and regular builds

are the responsibility of Anders Gebala, who will configure the build tool, write the code for the smoke tests and fix (or assign) eventual problems causing a nightly build to fail.

Experiences from Implementation phase 1

During the I1-phase the group mostly developed the different units that the final product will consist of. The integration of the different units had by the end of the I1-phase just begun and the regular nightly builds with smoke tests will begin with the start of Implementation phase 2, when more of the functionality between different modules will be developed.

Experiences from Implementation phase 2

Although we planned to get the nightly builds begun earlier during phase 2, we didn't get them to work until the end of implementation phase 2, due to some technical difficulties. Despite this the integration during phase 2 worked well, but as more and more features will be implemented, the risk of nasty integration problems occurring grows. The logs of the nightly builds are available at http://www.hut.fi/~agebala/pmoc/. We hope to soon integrate some tests to these builds as well.

Experiences from Implementation phase 3

Nightly builds worked well during phase 3. All our nightly builds were successful. The only problem we encountered was when HUT upgraded the operating system of the server (kosh) where we have our cvs and also our crontab-entry. The crontab-entry wasn't preserved so the nightly builds were done 2 times with old material.

The group did have some problems during phase 3. These problems were all detected the same day, and the member of the group who was the cause of the problem that time remembered to add the required file or files to the cvs after receiving mails with a subject like "a file is missing from the cvs".

Some unit-tests were also run in conjunction with nightly builds during phase 3. More information about the unit-tests can be found in the test report, section 4.1.

Experiences from DE phase and conclusions

Regular builds worked just as well during the last phase as they had during the other phases. During the entire project we didn't experience any unsuccessful nightly build, which I find a little surprising. Nevertheless the method worked as "quality insurance" during the project, working a bit like a kind of safety net. Why we didn't experience any unsuccessful builds can probably be explained with the fact that the project members usually worked together or simultaneously, and so any eventual problem could be "caught" as describe in Experiences from Implementation phase 3.

I would regard the use of regular build a good method to use in almost any software project. The gain outweighs by far the initial effort one has to put in to get a system for regular builds set up. The discipline of developing software is so complex and chaotic that every project needs any “safety net” or “quality insurance” they can get.

References:

[1] Pekka Abrahamsson et al.: Agile Software Development Methods: Review and Analysis, Espoo 2002. VTT Publications 478.

[2] Don Wells: Extreme Programming: A Gentle Introduction, http://www.extremeprogramming.org.

[3] Stephen R Palmer, John M. Felsing: A Practical Guide to Feature-Driven Development.

[4] David M. Dikel, David Kane, and James R. Wilson: Software Architecture: Organizational Principles and Patterns, Prentice-Hall, 2001.

[5] Martin Fowler, Matthew Foemmel: Continuous Integration, http://www.martinfowler.com/articles/continuousIntegration.html.

[6] Steve Loughran: Ant in Anger: Using Apache Ant in a Production Development System, http://jakarta.apache.org/ant/ant_in_anger.html.