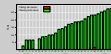


Introducing Automated Unit Testing into Open Source Projects

Christopher Oezbek

Freie Universität Berlin

The 6th International Conference on Open Source Systems, 2010-05-30 - 2010-06-02



Introducing Automated Unit Testing into Open Source Projects

Christopher Oezbek

Freie Universität Berlin

The 6th International Conference on Open Source Systems, 2010-09-01 - 2010-09-02

I would like to welcome everybody to this talk on “Introducing Automated Unit Testing into Open Source Projects”.

My name is Christopher Oezbek and this is work done as part of my PhD thesis at Freie Universität Berlin in the workgroup of Prof. Lutz Prechelt. First, I want to give you some background about this work. This work is part of the software engineering group’s effort at Freie Universität to analyze software processes and determine and implement improvements. In the domain of Open Source software development—which my PhD thesis work is about—we have concentrated on understanding how software engineering innovations can be introduced into Open Source projects.

This was driven by the desire to explain how volunteer organizations can achieve process change.

For this study we used an active case study methodology (similar to Action research), i.e. we directly interacting with an Open Source project to change their process, with the goal to learn about what would be important in this context.

- ▶ Goal: Explore Innovation Introduction in OSS
- ▶ Innovation: Automated Unit Testing via JUnit
- ▶ Target Project: FreeCol
- ▶ Prescribed Phases:
 - ▶ Lurk
 - ▶ Code & Collaborate
 - ▶ Withdraw
 - ▶ Observe

Introducing Unit Testing

└ Main

└ Study Design

Study Design

- Goal: Explore Innovation introduction in OSS
- Innovation: Automated Unit Testing via JUnit
- Target Project: FreeCol
- Prescribed Phases:
 - Lurk
 - Code & Collaborate
 - Withdraw
 - Observe

What exactly did I do?

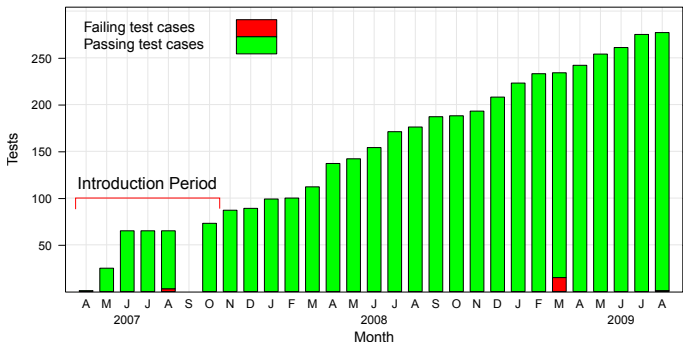
First, I chose an exploratory research question, with the goal to uncover interesting aspects of introducing an innovation into an Open Source project.

Second, as the particular innovation, I chose automated unit testing as a well-proven quality-enhancing practice.

Third, I chose a project from SourceForge's project of the month list, which was medium-sized and did not yet have any significant exposure to testing. The project I found was FreeCol, a game clone written in Java.

Fourth, I designed a four stage prescriptive process of innovation introduction, consisting of lurking, writing tests, collaborative with other one writing test, withdrawing and finally observing the project.

I conducted this starting April 2007, finished the introduction in October 2007 and collected data from the project in August 2009



- ▶ After the intervention more than 200 additional tests were created

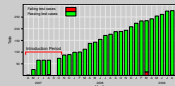
2010-06-01

Introducing Unit Testing

Main

Results I - Number of Tests

Results I - Number of Tests



- After the intervention more than 200 additional tests were created

As we can see, the number of test cases in the project has since the initial introduction increased linearly to reach 277 by the end of the observation period from 73, when I left the project.

Also the project has maintained the tests well with above 10% of their monthly commits dedicated to testing at a 95% confidence-interval, so that they are shown as passing in most months.

- ▶ The role of signaling in innovation introduction
- ▶ Learning behavior in OSS projects

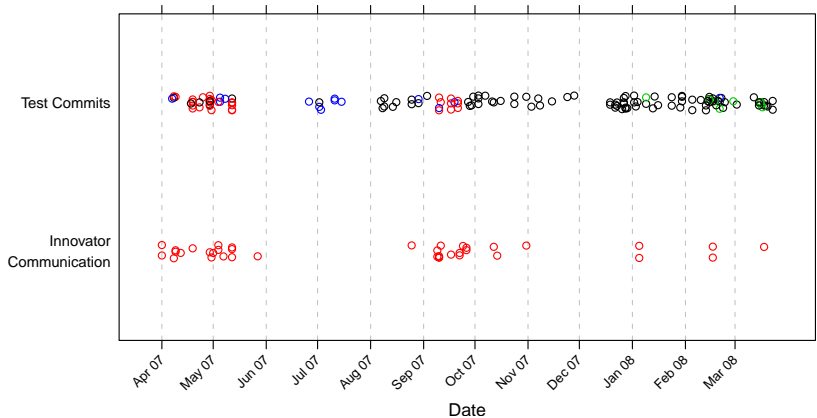
- ▶ Other results:
 - ▶ Test cases as a communication device
 - ▶ The role of the test-masters

- The role of signaling in innovation introduction
- Learning behavior in OSS projects

- Other results:

- Test cases as a communication device
- The role of the test-masters

1. From a quantitative perspective the introduction of unit testing was thus successful with many tests having been created and a base level of coverage established (people also reported liking testing), demonstrating the feasibility of an external participant establishing a new innovation. Yet what could be learned from a qualitative and conceptual perspective? There were in particular four things of which I want to discuss only two in detail because of the lack of time
2. I found that notifying others in the project is necessary to communicate status changes in participation - A concept I have called "signaling"
3. Observing the introduction of automated unit testing in the project for two years, led to a hypothesis about learning ability in Open Source projects.
4. There are more results in the paper and even more in the technical report. For instance, to explain the difference between increasing number of tests while coverage remained stable except for a few occasions the concept of a test master was created. Testcases were also used to communicate opinions and requirements in a more technically and formal way than using a mailing-list instead of just codifying existing specification into a run-time test.



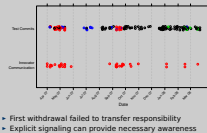
- ▶ First withdrawal failed to transfer responsibility
- ▶ Explicit signaling can provide necessary awareness

Introducing Unit Testing

Main

Signaling

Signaling



Let's move to the first result, "signaling" by looking at this strange gap in the number of test cases. First, here a graphical overview of the first year of the introduction to illustrate the point of signaling. First ignore the top. What you can see at the bottom is a timeline of the communication activity of the innovator. So whenever there is a dot in time, I as the innovator send an email to the project (in red). So you see that the introduction consists of two phases.

As you can see there are two main activity periods both from a communication and commit perspective shown in red.

The first peak represents the original introduction stages of lurking, activity, collaborating and withdrawing. But as you notice once I have withdrawn, there is a big gap of activity of around 6 weeks in which no changes are made to the tests and even then activity is too low to maintain the tests against the progress in the project. The test suite breaks.

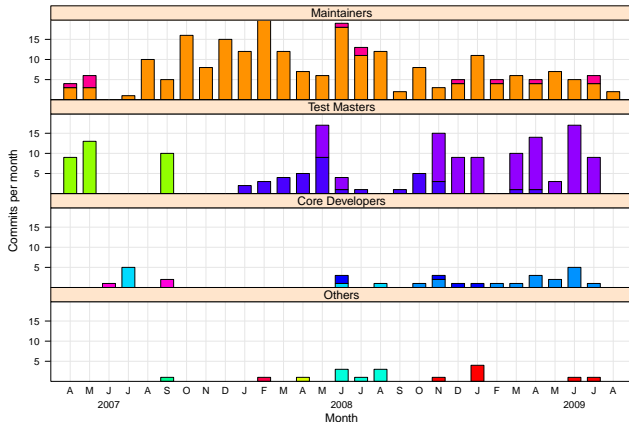
What had happened? It turned out that the strategy of withdrawing from the project over a time of only two weeks, had not induced the perception that I had left. Why? Because the lack of managing activity by participants (people come and go and rarely they discuss this) leads to a perception of presence in Open Source project. Even if I am inactive for two weeks, I will be still perceived as listening and maybe developing in the background.

Yet this of course makes the strategy of a withdrawal to hand-over responsibility invalid. Thus when I next returned to the project I signaled my departure after having fixed all test cases explicitly and as you can see this time the hand-over is much more gentle with the black developer taking over.

Take away message: (1) FreeCol (and likely other OSS projects) as well manage participation

implicitly. (2) This leads to false perception of presence. (3) Handing over responsibility for modules

or innovation introduction will thus require signaling to be efficient.



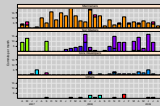
- ▶ Testing done by only a small proportion of developers
- ▶ Affinity to testing appears static over time

Introducing Unit Testing

Main

Learning behavior

Learning behavior



- Testing done by only a small proportion of developers
- Affinity to testing appears static over time

The second point I want to make is a controversial hypothesis and I hope we have some time to discuss it a little bit:

If you look at the following graph, which shows the number of test affecting commits per person over time split into four categories of project participants: Maintainers at the top, Test Masters next, Core developers below and all others at the bottom.

What is visible from this graph is that even though there are two maintainers, only one is doing all the test commits and even though there are 7 core developers, there are only 2 who have more than 10% of their commits affecting test-cases (deemed test masters here).

Why is that? What about the others?

The hypothesis, I would like to put forward here is that Open Source projects do have problems with knowledge acquisition and knowledge sharing.

It appears as if the tendency to do testing is not something people acquire over time, it is not a new skill they learn, but rather a project member is either a test-master or not.

This is obviously put bluntly, but think about it, which empirical evidence do we have that Open Source communities are great for learning about software engineering in contrast to learning how to work on a particular project?

1. Demonstrated the feasibility of introducing an innovation
2. Presence is informally managed \Rightarrow signaling recommended
3. Hypothesized: Learning behavior in OSSD is limited

Introducing Unit Testing

└ Main

└ Summary

Summary

1. Demonstrated the feasibility of introducing an innovation
2. Presence is informally managed \Rightarrow signaling recommended
3. Hypothesized: Learning behavior in OSSD is limited

In summary:

This study has demonstrated that it is feasible to introduce an innovation such as unit testing in a sustainable fashion into an Open Source project.

While doing so several interesting discoveries could be made for which it must be stressed that they are based on just a single case and thus certainly would need broad validation.

Of the four results uncovered, I have presented two to you

First, that presence in FreeCol was informally managed and thus changes in activity such as the innovator leaving needed to be explicitly signaled

Second, I have hypothesized that knowledge acquisition from the outside and sharing of knowledge internally to a project are less efficient than previously assumed.

I would have loved to discuss the other results as well, but I guess we rather move to the discussion. (The other results would have been: First, Open Source projects used unit testing to codify parts of communication, thus further strengthening the technical focus in Open Source development.

Second, we have described the role of a test master, as somebody who is very knowledgeable about testing and can advance the state of testing substantially.)

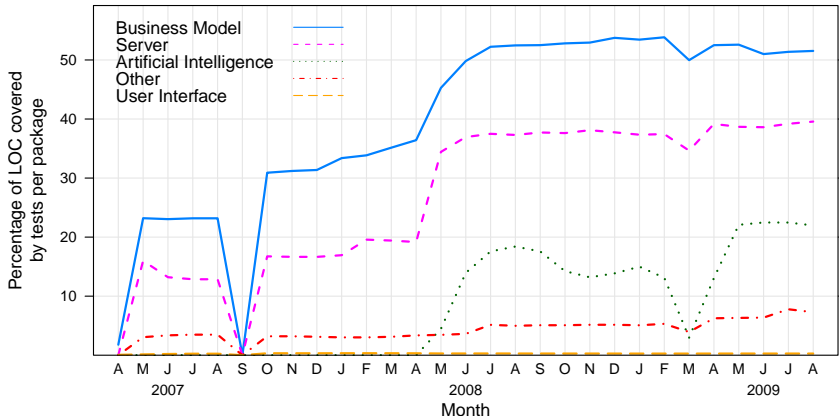
Thank you

christopher.oezbek@fu-berlin.de

Thank you

christopher.oezbek@fu-berlin.de

1. Thank you all for listening!
I'd be happy to take questions



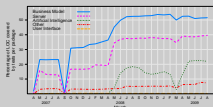
2010-06-01

Introducing Unit Testing

Main

Results - Coverage

Results - Coverage



Next, coverage is shown here over time per module. It is well visible that unlike the number of test cases there are only two major expansions in coverage (which in total reached 23% in total). One when I as the innovator introduced testing and the other one is here and was driven by a project member.