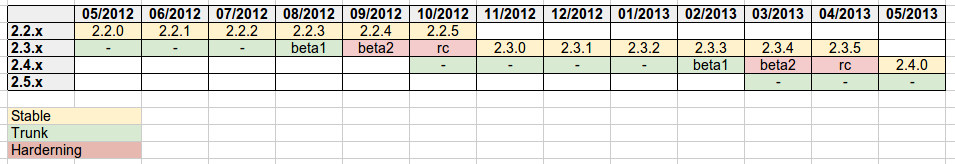
.Abstract(ss)

In the context of the subject ESOF we intend to analyze the software processes used by the team developing GeoServer, as well as the project management strategies(TODO:?) they use.

Regarding software process we will be going over how the different typical phases of most process models are/were(TODO:?) handled by the developers followed by an analysis of which models best fit what they have been doing.

.Project structure

GeoServer is an open source software server developed by hundreds of contributors. It follows a time boxed release model. For the stable branch minor releases occur every month and major releases every 6 months. After these 6 months the stable branch will become a maintenance branch and the unstable branch will become the next stable branch.



Most of the development is done on the unstable branch and is split into 2 phases: a 4 month open development period and 2 month hardening phase consisting of bug fixes and preparation for the next stable release.

The GeoServer project uses JIRA as their issue/bug tracking system.

In order to provide some control and leadership to the development of geoserver the developers use what they call a “Project Steering Committee” (PSC). The PSC is responsible for determining the direction the project takes, what changes are made to the core code, what major features are added and it also has the decision on any change that may affect the GeoServer community.

.Development process

>Specification and requirements elicitation

Changes to geoserver can be proposed by anyone as what the developers call GeoServer Improvement Proposals (GSIP). These proposals are then reviewed, accepted or denied by the PSC. Anyone can participate in the discussions over proposals. There is an attempt to keep close contact between users(clients) and developers, which can be the same. The development is therefore heavily prototype based. Most non critical changes result from prototypes from independent user forks that lead to GSIPs when mature enough, and eventually to pull requests. Once approved a GSIP will get a JIRA tracking number. This is how requirements are handled. Users propose changes, the PSC votes on them and they are added to a backlog similar to how Scrum works. Accepted proposed changes can be added to the release currently under development or be deferred for later dependent on the time it'll take to implement them, and developer availability.

Example rejected GISP:

https://github.com/geoserver/geoserver/wiki/GSIP-125---Layer-with-Service-Security

>Design and implementation

The project’s design is very collaborative and transparent to the userbase which can itself take part in the discussion. Even the core developers and PSC members have to go through GSIP process to make major changes.

A lot of the design discussions end up happening during the GSIP reviews and biweekly meetings of the PSC, which can be attended, upon request, by other active developers.

Example pull req with attached GSIP:

<https://github.com/geoserver/geoserver/pull/1182>

The various developers assigned to different tasks will have a high degree of freedom in terms of how they implement their assignments but there is a code review phase, that may be skipped for trusted developers on code expected to have smaller impact on the project, during which a developer’s implementation can be brought in line with what the core developers and the PSC want. The geoserver documentation provides extensive guidelines for any would be contributors and some tutorials on setting up to develop for geoserver, although some of the documentation appears to be outdated.

http://docs.geoserver.org/latest/en/developer/

>Validation

There are many ways through which users can voice their opinions on the state of geoserver, and they can, as has been mentioned, do it throughout the whole development process.

>Evolution

Because of its transparent development, which allows users to see how development proceeds, the way any end user can suggest improvements, report bugs, request features and even become themselves developers geoserver can very quickly respond to changes in user’s needs. The fact that anyone can access the source code and the branches for older releases also allows users for whom upgrading is impossible to adopt smaller changes and fixes.

>>Analysis

Agile software development is based on 12 main principles which we’ll now compare to what we have observed from GeoServer’s development process:

1. Customer satisfaction by early and continuous delivery of useful software:

GeoServer is setup so that every month there’s a minor release which allows them to deliver quick fixes and tweaks to their users very quickly

2. Welcome changing requirements, even late in development

The project actively encourages user feedback and involvement at all stages.

3. Working software is delivered frequently (weeks rather than months)

Refer to point 1.

4. Close, daily cooperation between business people and developers

This doesn’t really apply to GeoServer being a free and open source software. The community supporting the project, including the core developers end up doing whatever work there is on the business side of things so cooperation is a given.

GeoServer is an open source and free software, but there are many companies that use it and also that provide paid support and training for its users. As any other users they can participate in the development and some core developers work in fact for some of these companies.

5. Projects are built around motivated individuals, who should be trusted

The work for GeoServer is done on a voluntary basis, so there is very low risk of having developers who are “just going through the motions”. If anything, by its nature, the project is too closely linked to specific individuals, which could be problematic if they lose motivation.

6. Face-to-face conversation is the best form of communication (co-location)

The core team has biweekly skype meetings, but in presence communication is quite difficult again by the very nature of the project, which relies on people from all over the world to volunteer their time. The developers communicate mostly through e-mail. We did find that there was at least one development meeting among some of the developers in 2014.

7. Working software is the principal measure of progress

“The GeoServer Project recognizes that it is run those who are actually doing the work, and thus we want to avoid high overhead for ‘getting things done’.”

The quote above was taken from the geoserver documentation. Since the work being done here is on a volunteer basis making working software the only measure of success.

8. Sustainable development, able to maintain a constant pace

We have some doubts as to the team’s ability to maintain a constant pace due to the dependence on voluntary work, despite the main contributors seeming very motivated. There’s always the risk of more pressing issues disrupting the time they dedicate to the project before adequate replacements join the team. There is an effort towards a constant rhythm as evidenced by the release schedule and the fact that it has been adhered to strictly(for how long?)

9. Continuous attention to technical excellence and good design

The core team reviews the code and pull requests to ensure code quality is up to their standard, and through the PSC they also control the design and architecture.

10. Simplicity—the art of maximizing the amount of work not done—is essential

?

11. Self-organizing teams

Most of GeoServer’s contributors, including the core developers, do their development work on self-organized teams (most work seems to be done individually) on their personal forks, working on features and changes that they themselves have proposed.

12. Regular adaptation to changing circumstance

Refer to 1 and 2.

The way geoserver is being developed takes a lot of cues from the agile development principles. There is constant back and forth between developers and users, they respond quickly to change, development is geared towards prototyping both for core developers and other contributors, developers are self-organized and often self-sufficient prior to integration of their work.