Making a software factory

INF5750

What is a software factory?

- “A **factory** (previously **manufactory**) or **manufacturing plant** is an industrial site, usually consisting of buildings and machinery, or more commonly a complex having several buildings, where workers manufacture goods or operate machines processing one product into another.”
- A software business, or an open source project, is similar to a factory… It requires tools and processes.
- It’s worth thinking about software development like this

Main software factory tools

- Design tools
- Source revision control system
- Build and dependency systems
- Communication and documentation
- Bug reporting and project tools
- Test tools
- Continuous integration tools
- Delivery and deployment tools

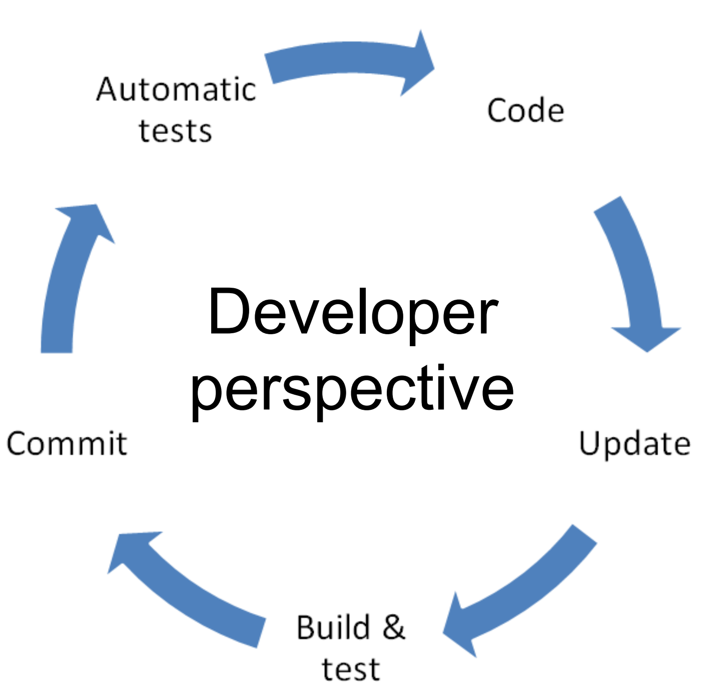
Continuous integration

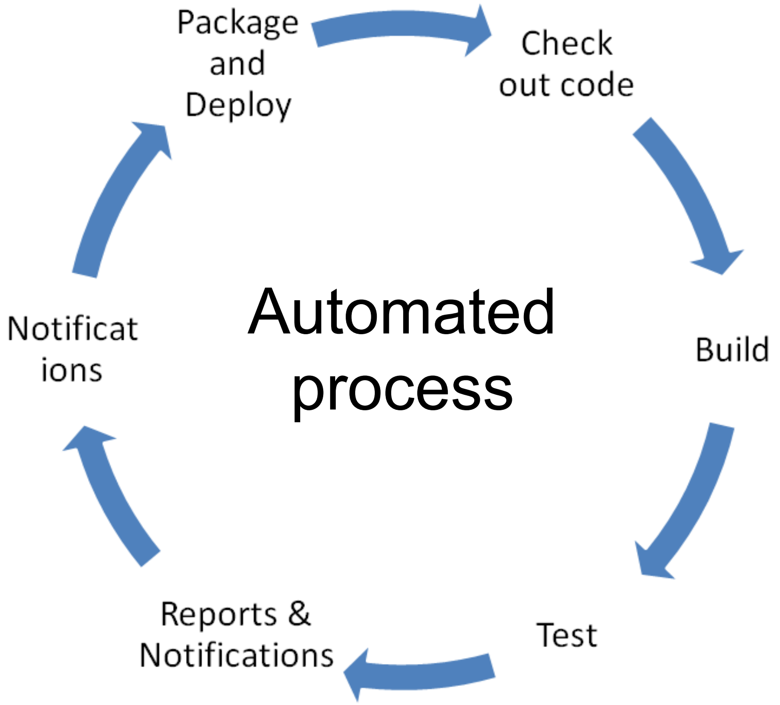
- Distributed software development
- Any developer can introduce bugs that can cause project failure
- Extreme Programming → CI
- Implement **Unit** tests, **Acceptance** tests and **integration** tests
- Integration tests check the integrity of the whole system, not just units
- Small code changes → system failure

Continuous integration

- The first step is to use source control and a build system that incorporates testing.
- Each developer should test before commit.
- There are lots of automated build systems that compiles, tests and deploys your software.
- DHIS2 uses Jenkins: http://apps.dhis2.org/ci/

Integration loop





Jenkins

- Basic stuff: builds, tests and deploys
- Helps you keep an overview of developers
- Notifies developers if their code breaks
- Tests reports, with history of broken builds
- Help keep track of deployed versions (file fingerprints)
- Extensive list of plugins. Source code management, build triggers, build tools, build wrappers/deployment tools, build notifiers, test reports…
- Deploy artifacts in the Maven repository

Some good tips

- Break up tasks. Commit often.
- Write unit tests, requirement tests and integration tests.
- Update your code from the repository regularly
- Implement peer reviews of source code
- Before committing code to the repository, always compile the entire (updated) code and run tests.
- Test in production environment - HW can wake bugs
- Automated testing does not replace manual testers
- Use your own products

Bug reporting tools

- Basic: report bug, get ticket, involve developers, discuss issue, see status, workflow, integr with other tools etc.
- For open source projects, often handled by Github, Bitbucket, Launchpad etc.
- Also standalone tools: Bugzilla, Apache Bloodhound, Jira etc. Blurred boundary to project management tools.
- Documentation, test planning, workflow, discussion around bugs etc.
- DHIS2 uses Launchpad’s internal features.

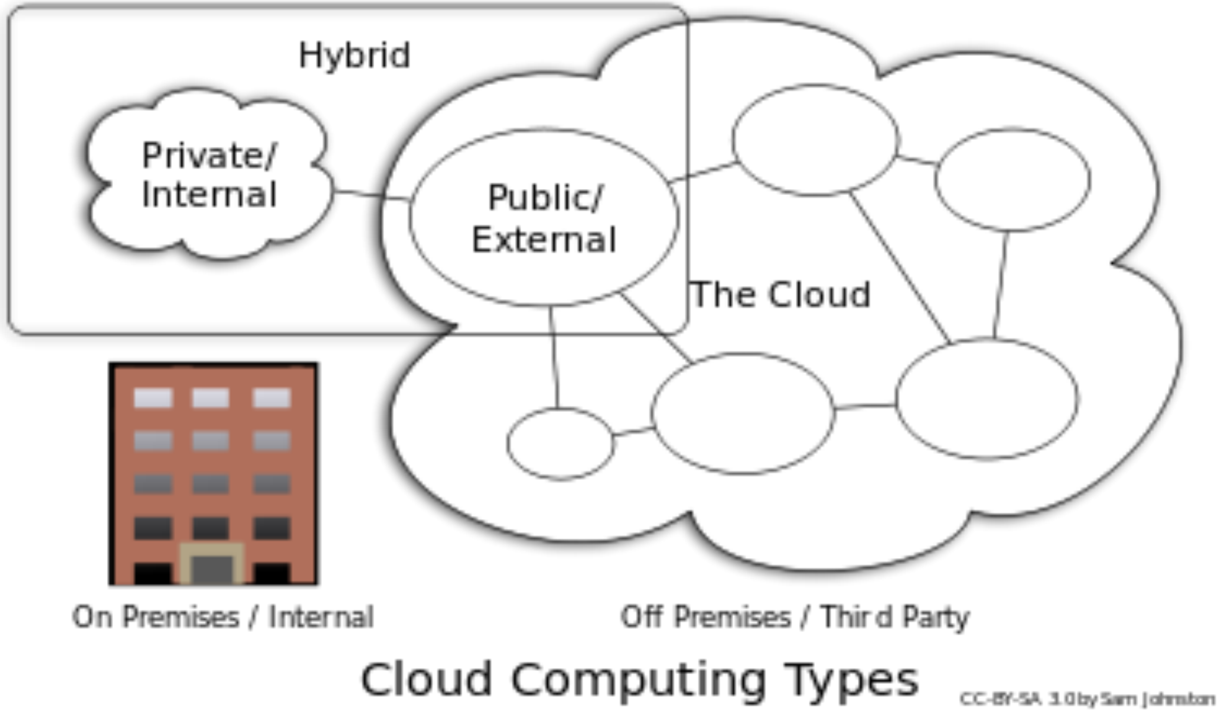
Documentation

- If source code documentation is done right, a lot of the documentation is done while coding
- Build systems generate nice-looking API documentation based on your code.
- Other documentation in open source is often Wiki-based (or other collaborative documentation software)
- There are free Wikis (Wikimedia etc) and commercial ones (Atlassian Confluence etc)
- DHIS2 uses Docbook. Group projects use Wiki

Deploying your software

- Moving towards cloud based deployment
- **Infrastructure** as a service (Linode, AWS etc)
- **Platform** as a service (Google App Engine, web hosting etc). More or less scalable.
- **Software** as a service (Salesforce.com etc)
- **Network** as a service
- Service deployment of your open source software has license issues. More on this later.

Cloud


Hybrid
Private/ Internal
Public/ External
The Cloud
On Premises / Internal
Off Premises / Third Party
Cloud Computing Types
CC-BY-SA 3.0 by Sam Johnston

Google App Engine (Cloud)

- Deploy your web based applications on Google
- Fairly high restrictions on what you can do. Not JavaEE.
- AppEngine Data Store, CloudSQL, Google Cloud storage
- Can run Hibernate and Spring, but has limitations
- OK for simple projects
- Free to start… pay as you go. Scalable pricing
- Your code is tied closely to Google’s system
- Deploy straight from Eclipse
- No ‘hardware’ concept…

Amazon Web Services

- Probably the largest and most famous P/IaaS vendor.
- Basic form: Get a linux installation. Full control. EC2
- More advanced: Use their APIs. Lock in to AWS.
- Also have Beanstalk, similar to Google App Engine.
- Pay as you go, scalable.
- Very high-scale platform and services.
- Free for one year with a very small instance
- Can use Hibernate and Spring etc to abstract APIs

Linode - virtual linux...

- If you don’t want to tie into a vendor
- Linode and others provide virtual linux servers. Has some level of scalability, but not as extreme as AWS
- You are sharing hardware with others. You have your own virtual Linux installations.
- Linode is run on XEN, an open source cloud system
- XEN is used by many similar vendors. Also alternatives to XEN.
- Another alternative - dedicated co-location/rack space