BITPlan

Open Tool Chains for Software Creation

Tools are discussed happily and quite often.

More often than not the tools determine the methods and not the other way round.

Tools promise to increase the productivity of a team but quite often the opposite happens.

Tools can be used as an instrument of power if access to them is blocked by license or know-how issues.

- How did the tool chains for software creation evolve as a means to bridge from whishes to delivered software?
- What are common reasons for this bridge to fail / the tool chain to break?
- How full traceability for every bit of your software is the key to a successful tool chain.
- What does a typical set of software development tools look like these days?
- The “git” – story: Why are open tool chain approaches so successful
- What are people expecting from a tool chain and it’s supplier?

Speaker: Wolfgang Fahl

- Wolfgang Fahl
- Diplom-Informatiker RWTH Aachen

CEO and owner of BITPlan GmbH

Wolfgang Fahl is Diplom-Informatiker and CEO of BITPlan GmbH. He has been working in software engineering for 25 years.

Software quality is particularly close to his heart, since he helped develop software for life-sustaining systems. Since 1996, Wolfgang Fahl is serving actively as a consultant and entrepreneur for organizations that develop software so that the result “fits”.

BITPlan

FBTT

FBTT=„From Brain to Terminal“
is still the most successful software creation tool chain “

As long as the software is written by the person also specifies the requirements for the software the FBTT approach is possible. The main tool is then the human brain. The quality of the result depends entirely on the skill of the "hacker" which converts the software on the basis of his ideas directly from your code.

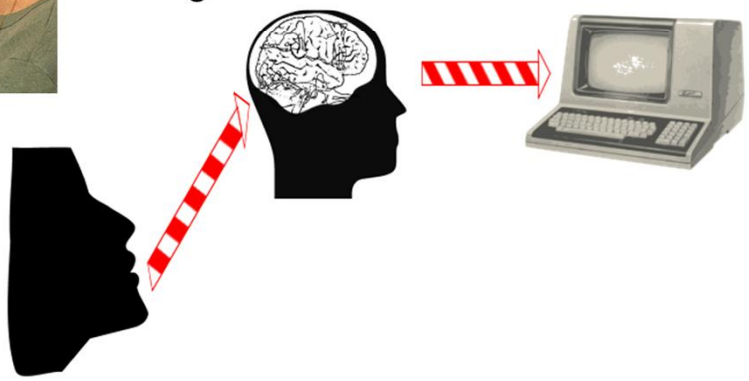
The lone programmer must be familiar with the subject area for which the software is to be created. Therefore for domains outside of the computer science disciplines the only suitable people are those which have acquired programming skills in addition to their domain knowledge.

A number of successful commercial software for different industries arose in this way. It is interesting that the better understanding of content has often enough impact to more than offset the worse programming skills in comparison to professional computer scientists. So quite a few industry solutions from in the last century where initially created in this way.
Only with time and increasing success in the market there was a need to expand the software development process and improve.

BITPlan

Twobraintoolchain

As soon as someone starts “whispering” the requirements to a second person the tool chain gets longer:



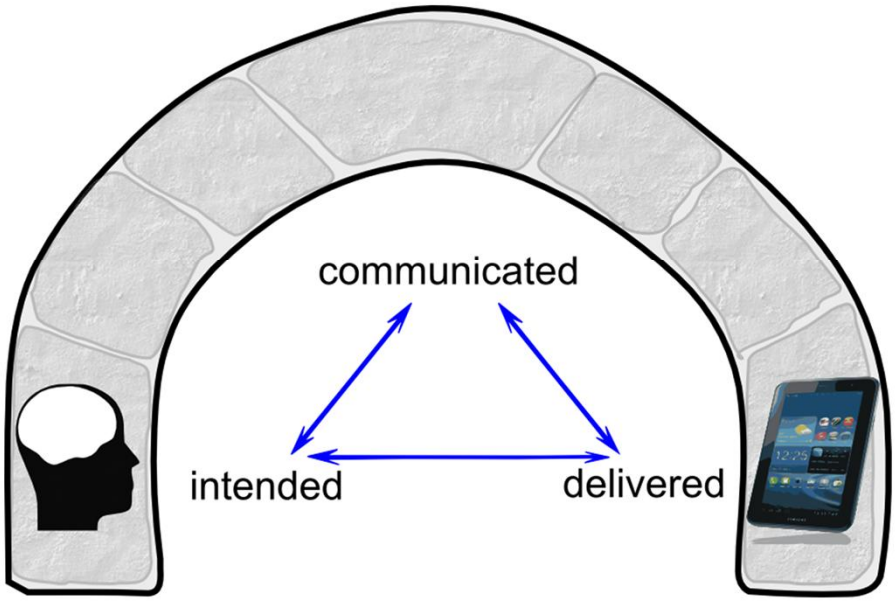
As soon as two or more people are involved in software development, it is no longer ensured that the requirements are actually implemented as intended.

The dilemma of human communication is at work here.
"That I have not said" and "You simply can not understand me" are some popular book titles about human communication illustrating this point.
From the point on where a two- or more-brain tool chain is applied all the problems arise that stem from the distribution of knowledge.

The binding state of the software is still defined by what is in the source code.
Only when changing the programmer it is conceivable whether this approach will be feasible in the long term.
Software created on a contract basis is still very often produced today by a two-brain process.

BITPlan

The Bridge



What is defining the quality of software?

- How well does what is communicated match with what is intended?
- What is the best possible match between what is intended and what is communicated?
- How well does what is delivered match with what is communicated?
- What is the best possible match between what is delivered and what is communicated?
- How well does what is delivered match what is intended?
- What is the best possible match between what is delivered and what is intended?

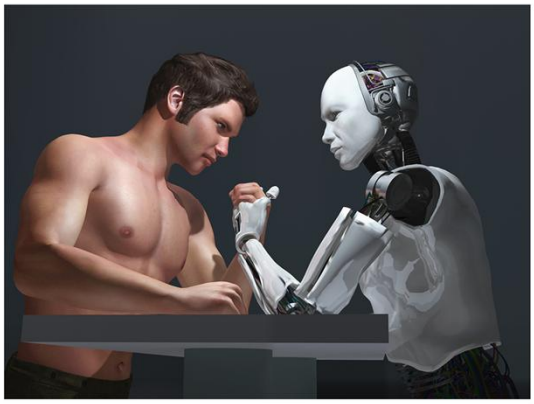
The activities on the left side of the bridge are about finding out what the software should do and how the software should do it.

The activities on the right side of the bridge are about the technical implementation of “what is communicated" in bits and bytes and then finally delivered this software.

With the increasing complexity of software and the organizations developing it, the tasks at hand need to be distributed between more and more brains and the need for methods and tools increases.

BITPlan

Man or Tool?



- Usability
- Availability
- Traceability
- Cooperation
- Performance/Speed
- Reliability
- Cost

…

What is the best way for passing over the bridge? After it had been established that is not so easy to create software that stakeholders are satisfied with the result, there were some suggestions:

- process Improvement
- usage of methods
- usage of tools

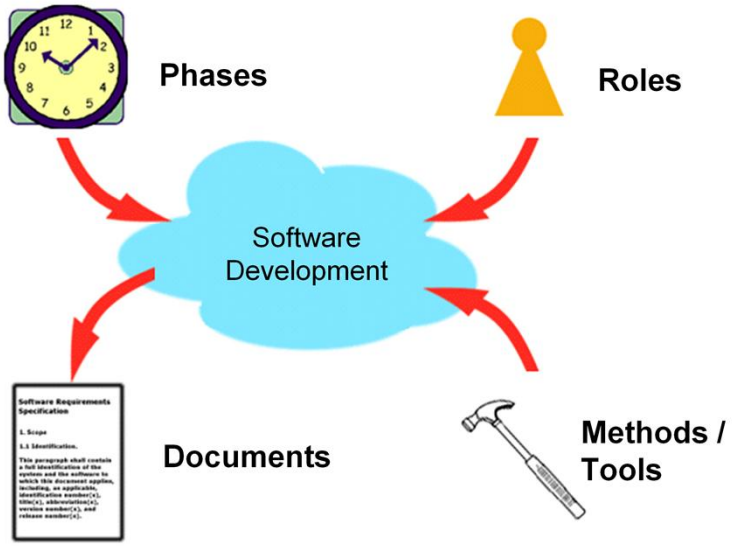
each pursuing the objective to improve the software development. The typical quality attributes of the final software product can also be applied to the way over the bridge.
The agile manifesto states:

"… We have come to value… Individuals and interactions over processes and tools" – " … That is, while there is value in the items on the right, we value the items on the left more. …"

see http://agilemanifesto.org

The provoking "Man or tool" argument as shown above or even "man vs. tool" I personally think is overdone. I personally do not subscribe to the claim of the English Wikipedia "CASE" article.? "CASE tools were At Their Peak in the Early 1990s" either.

BITPlan Software-Development Process



Many ideas have been invented, how software can be developed in the most well organized manner. Most of these ideas make use of the above terms:

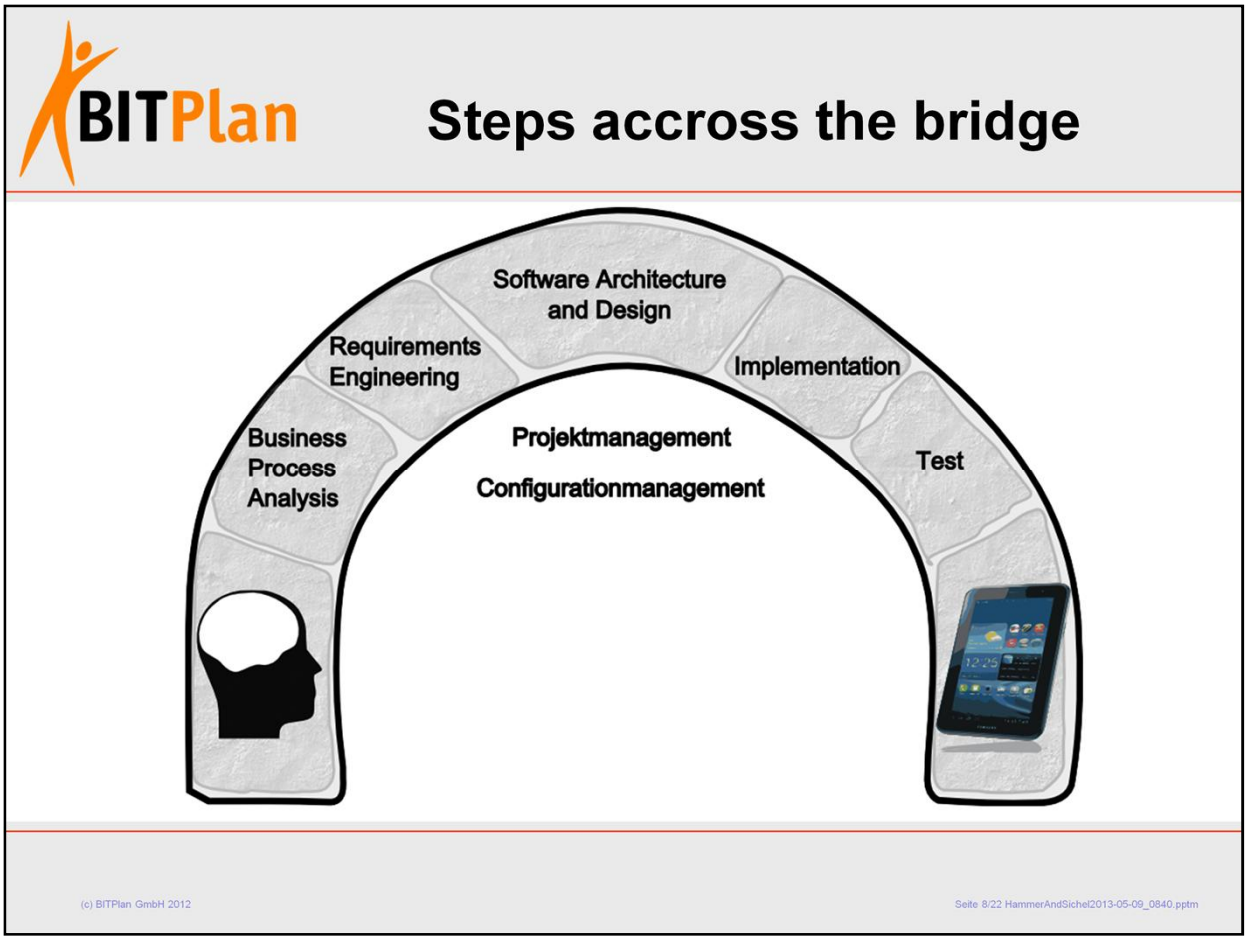
- Phases: the time distribution and decision about the sequence of activities
- Roles: the division of labor for the work items involved
- Documents: establishing the required result types
- Methods / tools: the systematic and (semi-) automated processing

The corresponding Wikipedia article
http://de.wikipedia.org/wiki/Liste_von_Softwareentwicklungsprozessen
lists over 30 different process models for software development.
Accordingly, the list of possible methods and tools is considerably longer.

See also

http://en.wikipedia.org/wiki/List_of_software_development_philosophies

with an even longer list of principles that are suggested to be applied.



The example shown above shows five steps across the bridge with two supporting cross-cutting functions. In total there are 7 roles to be filled. For all of these roles and functions, there are specific methods and tools

Steps accros the Bridge

Methods:
BPA, de Bono, Five Why's, MoSCoW, PESTLE, HEPTALYSIS, MOST, SCRS, SWOT, VPEC-T, …

Standards:
BPMN, EPC/EPK, …

Tools:
Abacus, Adonis, ARIS, Cameo, ezBPR, Fujitsu ABPD, iGrafx, Abacus, Pegasystems, Corporate Synergy, Visio, Innovator, …

Vendors:
IDS Scheer, IBM, Metastorm, iGrafx, Mega, Casewise, Microsoft, EMC, Sybase, Lombardi, Business Genetics, Savvion, Tibco Software, Ultimus, Sparx Systems

As an example, lets for once take the "first building block" of the bridge and examine it (in a non exhaustive way).
It is readily apparent that the time that is available for this presentation is by no means sufficient to look at the entire landscape of software engineering tools and methods possible into this level of detail.

This is fortunately not what the presentation is about and is stated her only to adjust expectations a bit.

Tools and People

- We have no tool
- I have a tool
  and you have none
- I know how to work with my tool and you do not
- I have a license and you don not
- I have one tool
  and you have another tool

No matter what tools are available, the use or non-use of tools leads to some standard problems. A small but important selection is shown above.
Tools only make sense if they are available, understood and accepted. The Cockburn test may be applied in this sense not only for methodologies but it is also valid for tools:
See Alistair Cockburn, Agile Software Development, p 145:

- The project was delivered ..
- The leadership remained intact
- The people on the project would work the same way again

I.e. the tools must have been really productive and worked with in to useful way.
The project or the organization / group has "survived" the use of tools - such as it is not gone bankrupt.
The parties would voluntarily use the tool/the tools again the next time.

The broken bridge/tool chain

- Reduction of Productivity, useless results
- Failure of cooperation
- Disjoint results
- Island solutions, dead ends

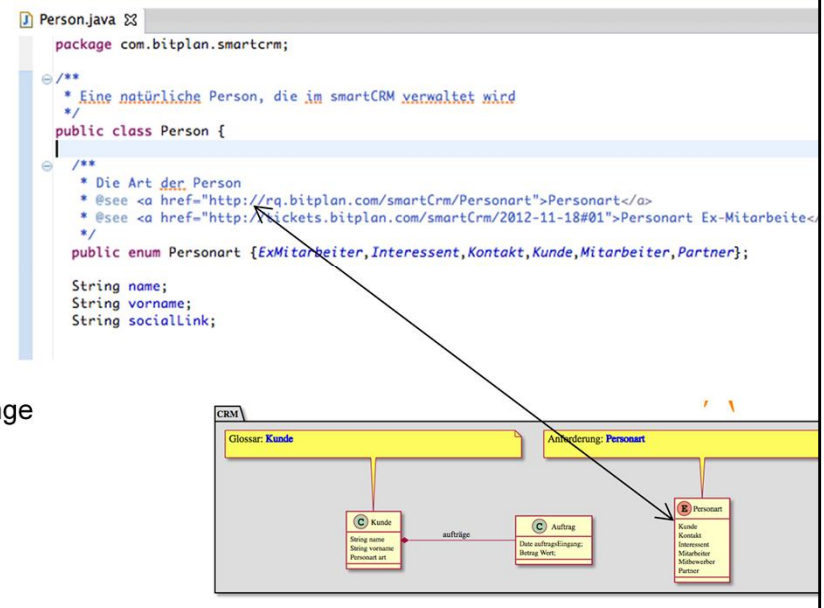
When it comes to breaking it can be applied to the bridge metaphor as well as the chain metaphor.

A broken bridge is as useless in connecting two sides as a broken chain is.

The usage of tools can cause the opposite of what is intended. This effect is even stronger with tool chains than with individual tools. These negative effects must be avoided.

Traceability

- Requirements
  - Open Questions
- Model
- Source Code
- Documentation
- Test cases
- Project plan
- Issues
  - Bug Reports/Change Requests
- Support
  - Support Requests
- …



An important criterion for putting together a tool chain is the assuring the traceability of results. Superficially, one might think that it is especially important looking forward to the from requirements to the finished software (in terms of the bridge ie from left to right). But actually many software projects, suffer from the dilemma of not knowing which part of the software can be omitted if a specific requirement is waived.

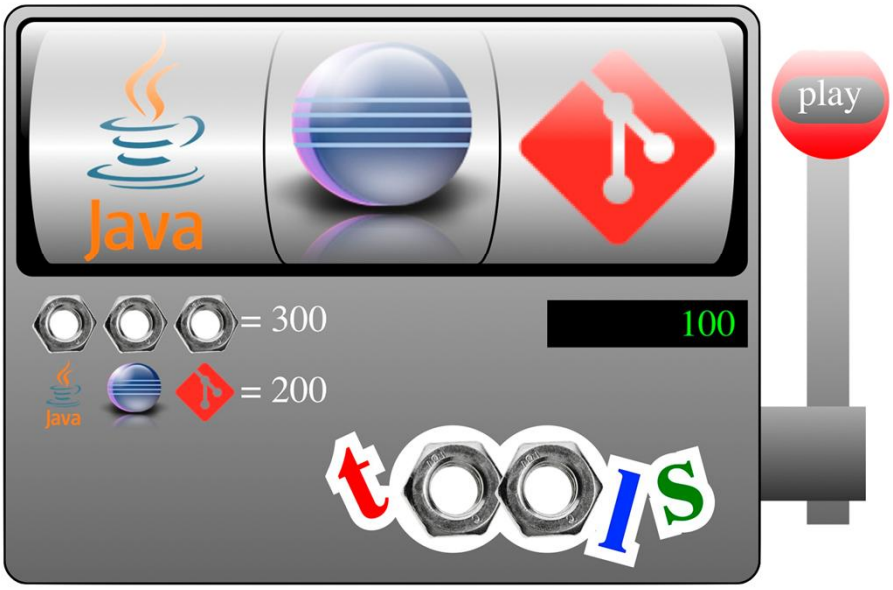
To solve this dilemma it is also necessary to ascertain the trace backward over the bridge and find out which **other** requirements may also ask for the bits that might be superfluous know? As long as this forward-backward (bidirectional) traceability is not applied, a typical course of software projects will be repeated continuously: after some time the software is declared to be unmaintainable and redeveloped from scratch. It can be assumed that a well design tool chain could help to avoid this situation?

See also http://en.wikipedia.org/wiki/Traceability
To enable the traceability of software architecture documents, it is necessary to use unique cross-references between different results.
Each results needs to have a unique identifier. Ideally, an Internet / Intranet approach is used to accessed the individual results via regular html hyperlinks?.

The example above shows an example of a hyperlink from the source code to the requirements management system. (Eclipse - smartGENERATOR / ACGenerator -> smartRQM)

BITPlan

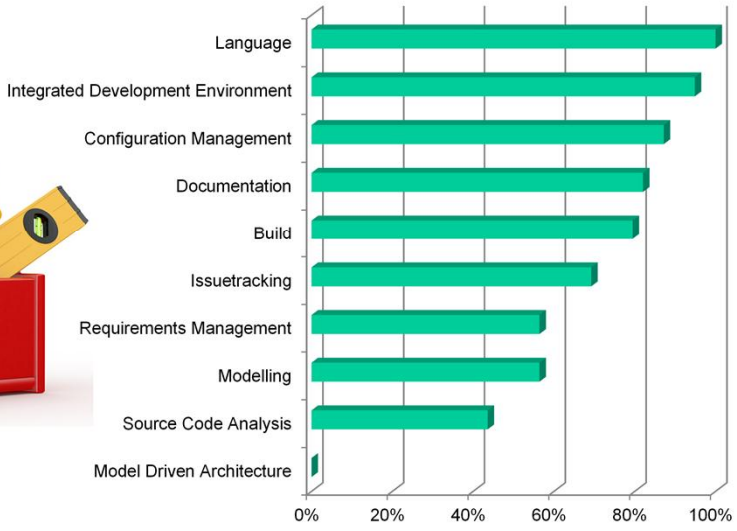
Tool selection



In fact, the market for tools is dominated by oligopolies.
So the tool selection is easy if you limit yourself to the top dogs in the sense of "nobody ever got fired for buying IBM".
In fact, only very large organizations can afford to develop their own software development environments. This is somewhat reminiscent of the time when it was clear that the development and commercialization of operating systems themselves can fail, the above still so admired IBM and its O2/2 failure being a prominent example. With Eclipse, IBM has made a significant contribution to the IDEs.

It is amazing that Microsoft is still the market leader with Office in the requirements management tool segment.

Tool survey

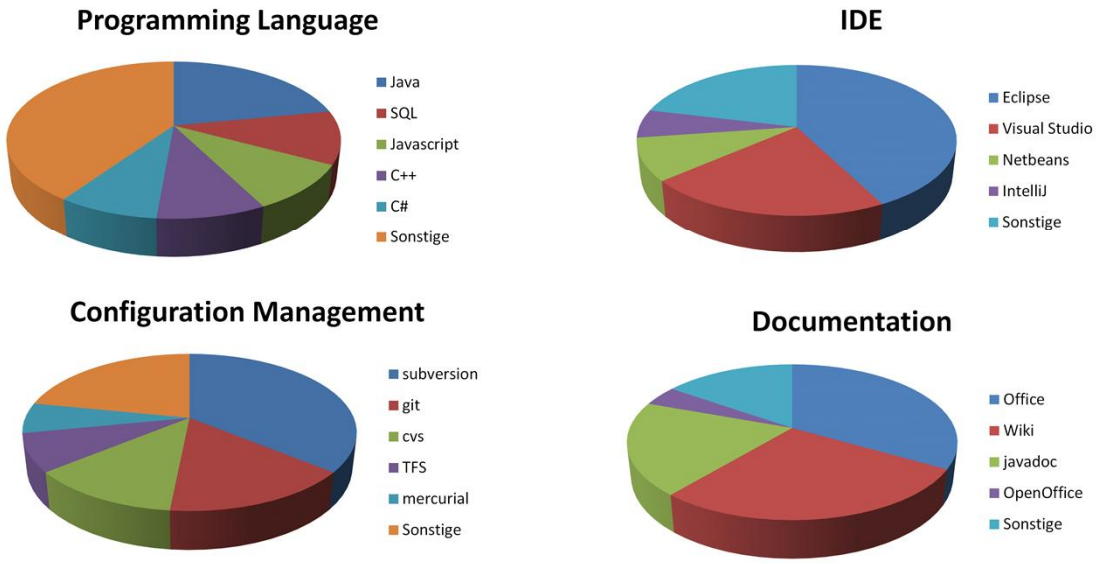


Tools take a crucial role within the software development process. Some tools such as the compiler and editor are indispensable, others only more or less useful. The above statistics on the use of different types of tools are based on a survey. The survey was first started among the participants of BITPlan CPSA-F students and later published online to let more people take part in it. So far some 60 organizations have filled in their tool chain setup. It shows that the tool chains are getting more and more complete, even if the proportion of organizations with significant gaps in the tool chain is still quite striking.

The list of requested tools was limited to 10 - test tools in particular were not queried. Indeed, there are of course many more tools. In particular, if we would add any IDE plugin for each type and each utility this would result in an extremely vast number of possible tools.

BITPlan

Oligopolies



For the most important tool categories there are only a few market governing vendors.

BITPlan

The „git-Story“

- 2005 CM „stopgap solution“ for Linux Kernel Sources
- 2008 GitHub:
  3.5 million users
  (+1 M/4 month)
  6 million repositories
  (+1 M/3 month)
- Success factor:
  complete(!?)
  tool chain?

GitHub has grown in 5 years (2008-2013) to 6 Million Repositories. In comparison the Portal “SourceForge” grew to 250.000 Projects in 10 years (2000-2010).

See also http://en.wikipedia.org/wiki/Git_%28software%29

http://www.zdnet.com/github-celebrates-fifth-birthday-3-5-million-users-and-six-million-repositories-7000013883/

Co-founded by Tom Preston-Werner, Chris Wanstrath and PJ Hyett, GitHub now employs 158 'hubbernauts'.

GitHub raised $100m in funding last year from Silicon Valley venture capitalist firm, Andreessen Horowitz.

BITPlan

The git „toolchain“

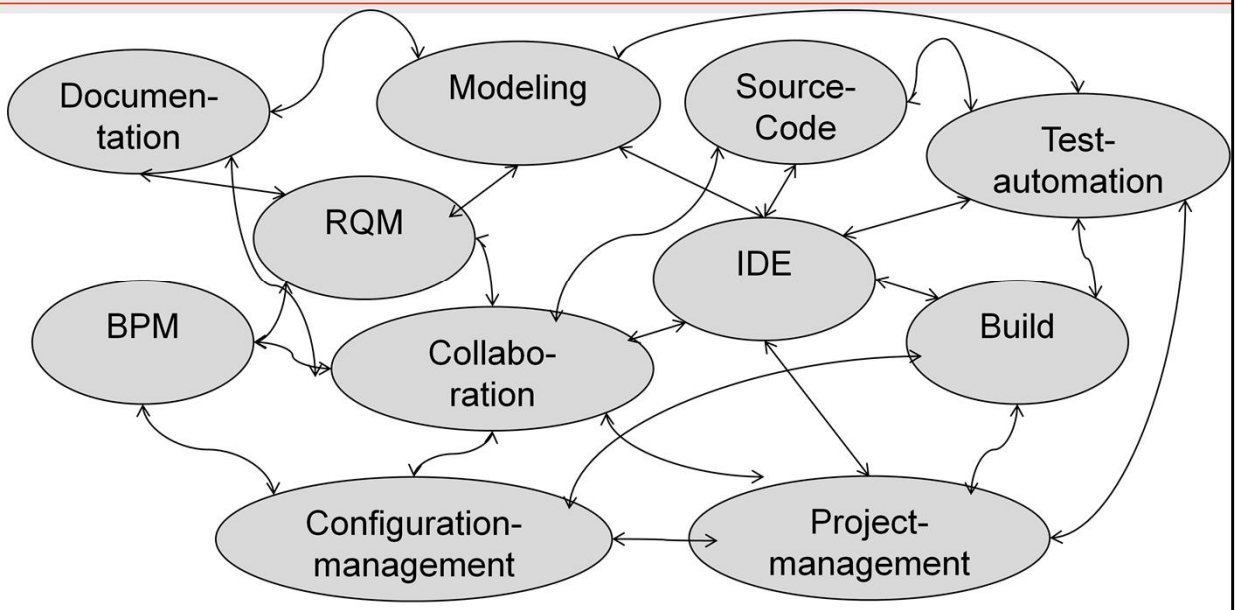
- Configuration Management
- Issue Tracking
- Webhosting
- Social Organization

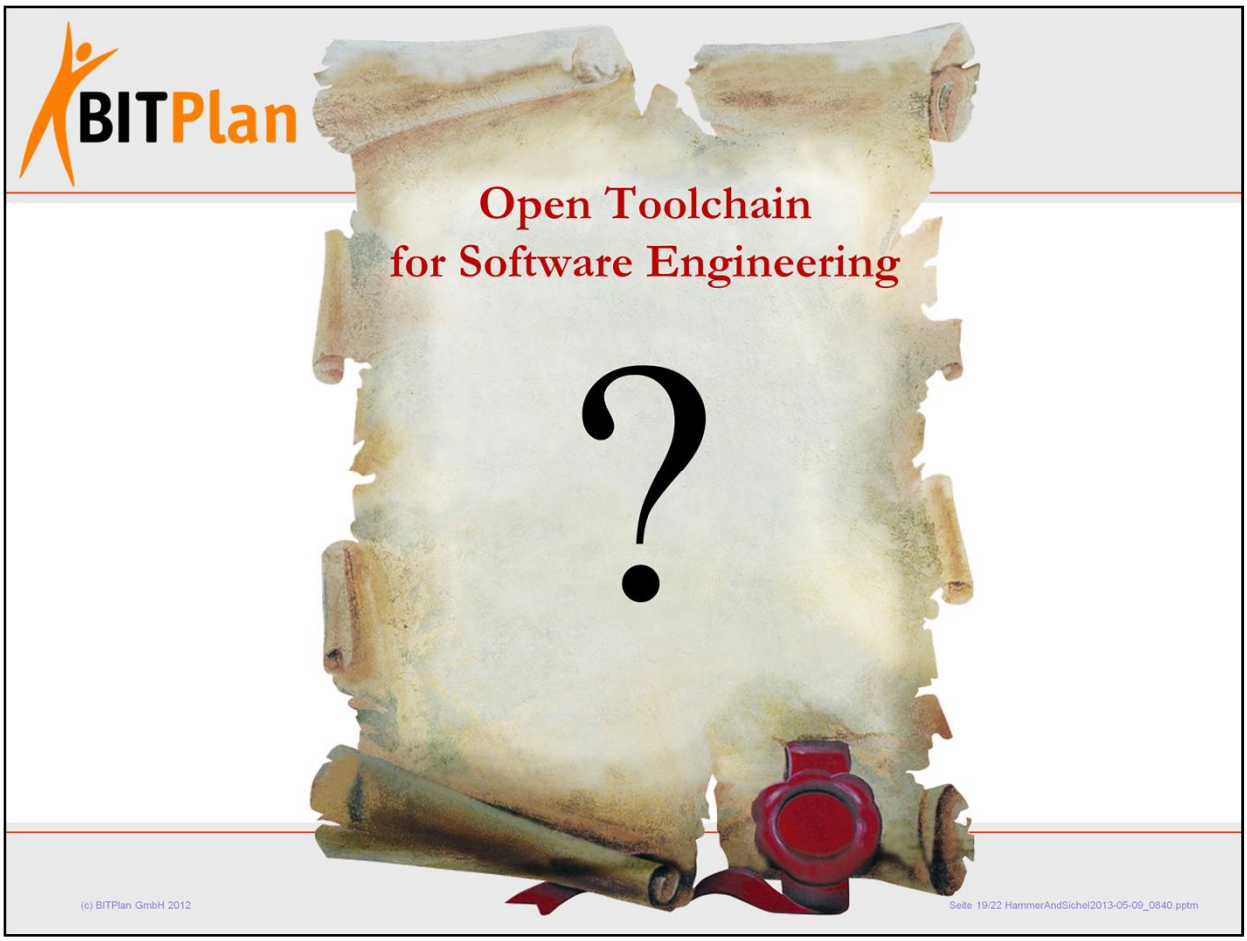
All other tools are considered to be naturally/readily available on Linux

Surprisingly, the collection offered by github tools is relatively straightforward, and even in some sense smaller than that of competitors. However, the success of github is greater if measured by the number of users / repositories.

BITPlan

Toolnetwork





How would a manifesto for an open tool chain for software engineering look like?

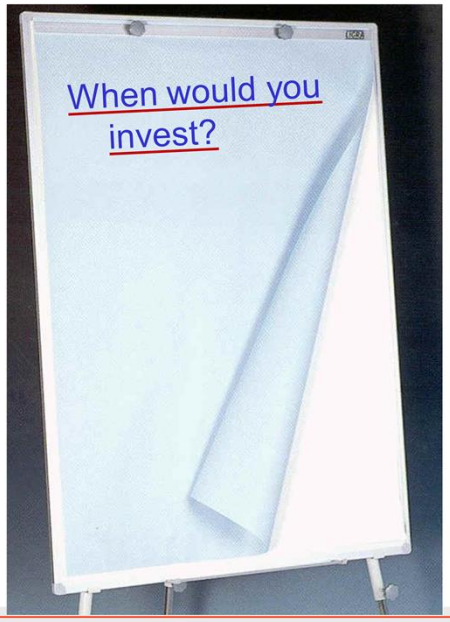
Important points are, in my view:

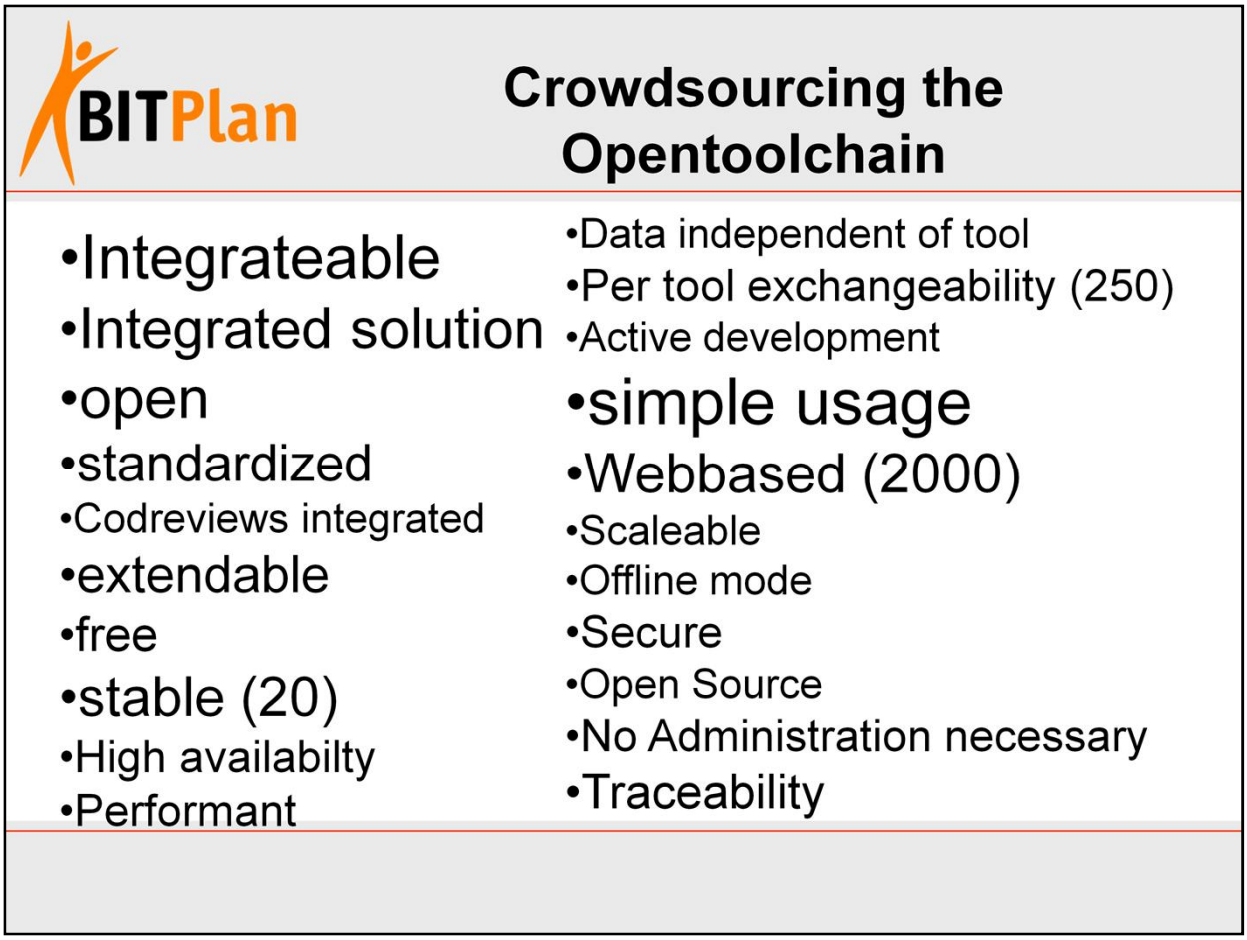
- Tools must be understandable that does not only mean that they are easy to use, but that the underlying concept does not introduce unnecessary complexity.
- Tools must promote cooperation.
- Tools must provide results which can be connected and therefore traceable. The links need to be stable and work in both directions. There is not necessarily a cause-effect relationship, but only a link
- Tools must be useful, that is promote productivity, the investment in effort and cost for the use of the tool must be nicely returned by the benefits they bring with.
- Tools must be available and accessible. I.e. There should be no technical or licensing technical barriers to the use of the tool.
- Local, Intranet and Internet use must be equally possible with no restrictions imposed by the mode of access.
- Tools must work together - the responsibility for the co-operation capability lies with the tool manufacturers not with the tool users.

BITPlan

Crowdsourcing the Opentoolchain

What properties should an open tool chain have so that you would invest in the organization that created it?





From some 60 participants:

Size of font:

24 Points = 1 Vote

Per extra vote font size increases by 2 points

(amount in EUR that people would invest)

Questions / Discussion

www.BITPlan.com

Wolfgang.Fahl@BITPlan.com