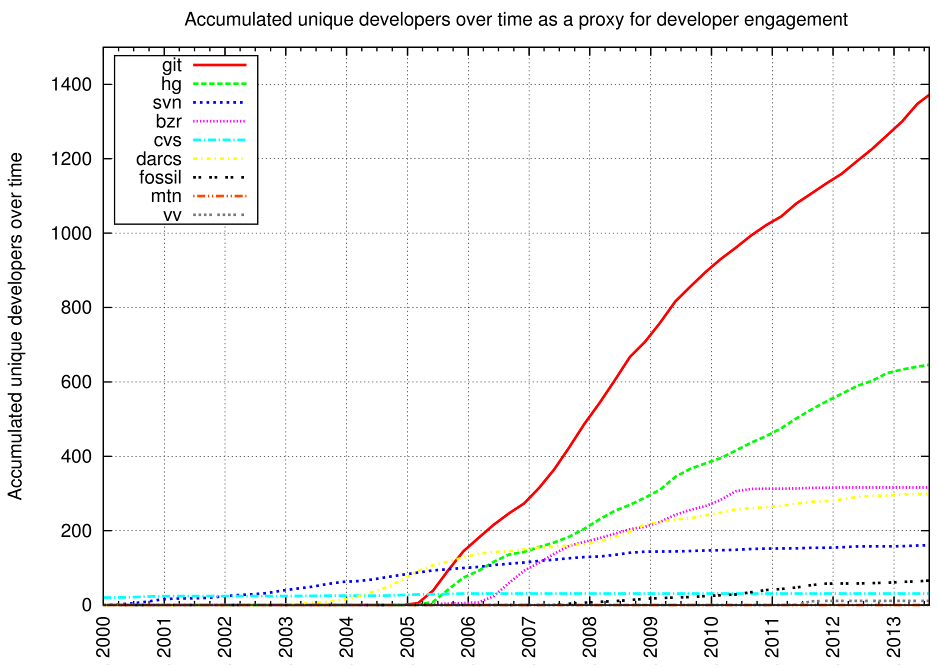
switch::gears



# SCM Ranking, Q3 2013

Number crunching by Knud Poulsen & Aske Olsson

Switch-Gears ApS
info@switch-gears.dk

## Contents

## 1 About Switch-Gears

Switch-Gears ApS is a software development productivity company based in Copenhagen, Denmark, spun off from Nokia's closure of its Copenhagen Research and Development site.

We provide training, migration, best-practices and bespoke development in the software tools and processes field (aka. "DevOps" or "Continuous Software Delivery"). We specifically focus on the highest innovation rate, highest ROI, continuous delivery stack: Git SCM, Gerrit Code Review, Jenkins CI, Redmine, etc.

We help our customers transition smoothly from legacy to leading-edge tools, then we help them make the most of those tools; with offerings such as installation, hosting, configuration, on-site, phone and e-mail support, upgrades, refresher courses, in-depth trainings and custom plugin and script development.

This and similar tools ranking analyses are intended to help Switch-Gears customers make the quantitatively best, most well informed, decisions within the software development tools landscape.

For an informal chat about how we can help improve your software development activities, please don't hesitate to email us at info@switch-gears.dk or call us on +45 20819928.

Also be sure to check out our latest product offerings on gitgear.com.

## 2 Introduction

SCM (Software Configuration Management) selection is arguably the most important tooling decision you can make as a software development team. SCM is the one tool that sits at the center of your entire software development process.

Whether you are a developer tasked with implementing a new feature or hunting down an illusive bug; a manager tracking development progress against requirements; or an operations engineer working towards automatically testing and deploying your applications: you are constantly leveraging and interacting with your SCM system.

There are many considerations when selecting an SCM system, but the key questions are simple: is the system technically up to scratch?, will the system be around in 10 years or will you need to change it again soon?, is there a large online community so your users can support themselves via online search?, are you able to hire pre-qualified staff?, will the tool develop advantageously over time allowing improved workflows and processes? Is it easy to migrate again in the future, when a new and better SCM system emerges?

The ranking criteria outlined in this study are not boilerplate "feature XYZ vs. feature ABC" comparison lists and charts. Such side-by-side feature lists are useful, but also short-lived and fragile, especially when the tools are evolving at breakneck pace. Instead we go to the root of the problem: the tool users and tool developers. This approach is more akin to modern marketing metrics like net promoter score and similar community engagement metrics. So the questions narrow down to: What does the development effort look like?, What does the user engagement look like?, How many surrounding and supporting tools are available for the system?, What does the sunk cost investment look like?

In our experience applying these "network effect analyses" rapidly and effectively narrows the list of selection candidates for any given class of tool, to 2-3 serious contenders and one clear frontrunner. This is also the case for SCM.

The network effect argument is simple and intuitive: "the more people use, develop and extend a system, the more valuable it becomes", conversely: "the more valuable a system is, the more people will use, develop and extend it", i.e. a positive feedback loop. This is the same positive feedback loop underlying the rapid growth and dominance of Google for searching the web, Jenkins for CI, Facebook for social networking, etc.

## 2.1 Application of network effect analysis to SCM systems

Several network effect indicators are used to evaluate the SCM systems on two key axes.

**User engagement**

1) Google trends - search volume as a proxy for user engagement.

2) Mailing list traffic - monthly mailing list volume as a proxy for user engagement.

**Developer Engagement**

1) Code activity - accumulated code changes over time as a proxy for developer engagement.

2) Developer renewal and diversity - accumulated unique developers over time as a proxy for developer engagement.

3) Secondary developer engagement - GUI's, frontend's, surrounding and integrating tools and systems.

4) COCOMO cost modelling - to get a rough idea of the investment sunk into each system.

## 2.2 SCM systems investigated

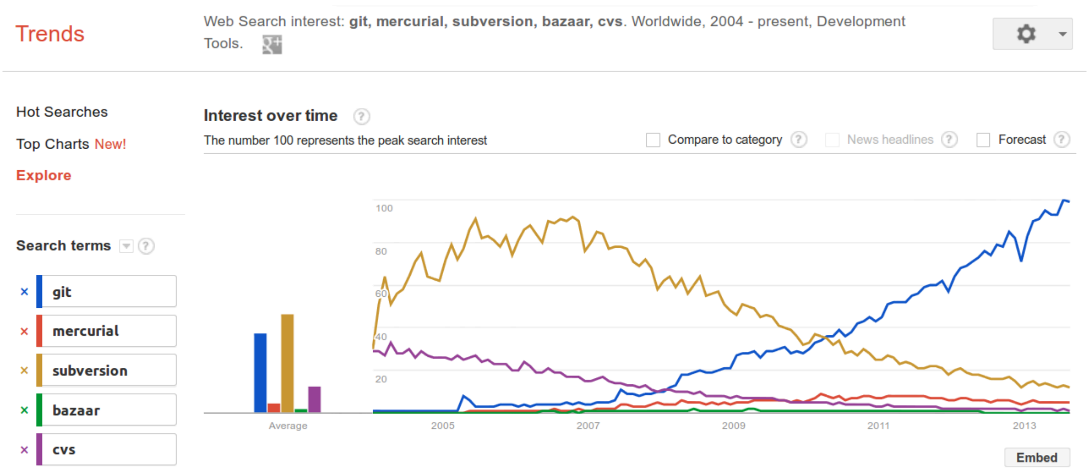
All current open source SCM systems are investigated, most are not worthy of serious consideration and are included solely for comparison and context. The following list enumerates the candidates, including overall architecture (distributed vs. centralized) and main development languages. In this and the following charts and graphics the three main contenders (git, hg and svn) are listed first, after them the listing is alphabetical.

| SCM System | Architecture | Main dev. languages |
|---|---|---|
| | | |
| Git | Distributed | sh, c, tcl |
| Mercurial (hg) | Distributed | python, sh, c |
| Subversion (svn) | Centralized | c, python, java |
| Bazaar (bzr) | Distributed | python, sh, c |
| CVS | Centralized | c, perl, asm |
| Darcs | Distributed | sh, c |
| Fossil | Distributed | c, tcl |
| Monotone (mtn) | Distributed | lua, cpp, sh |
| Veracity (vv) | Distributed | js, c, cpp |

## 3 User community engagement analysis

### 3.1 Google trends

Google trends search volume over time within the “Programming/Development Tools” category.



Web Search interest: **git, mercurial, subversion, bazaar, cvs**. Worldwide, 2004 - present, Development Tools.

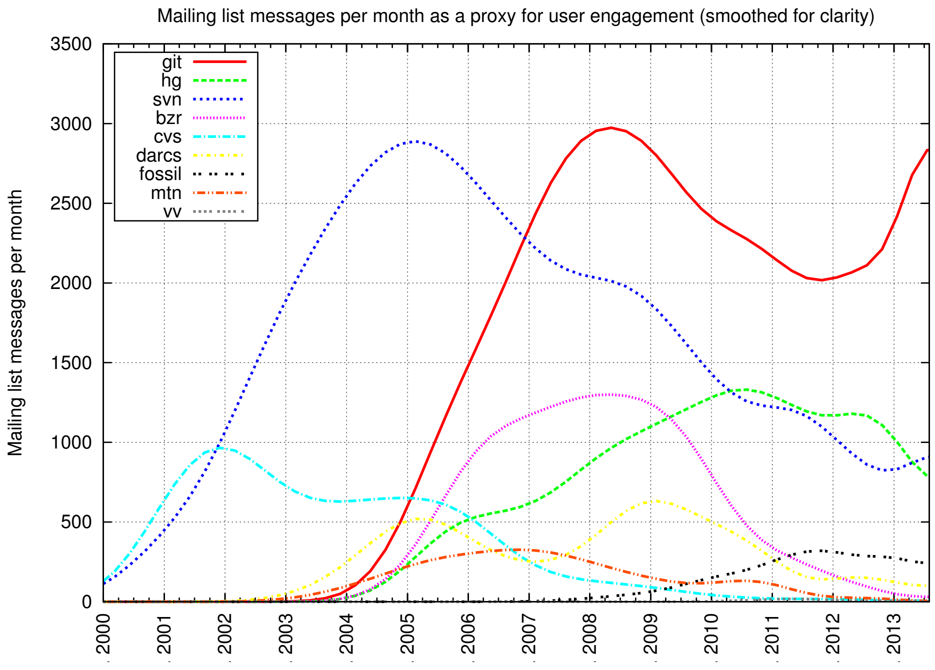
**Interest over time**

The number 100 represents the peak search interest

The conclusion from this graphic is clear: Subversion is rapidly being ∼1:1 replaced by Git within the software industry as a whole. Another logical conclusion is that if you are selecting a tool based on minimal support load going forward, i.e. leveraging the end users ability to support themselves through online search, then Git is the preferred option.

### 3.2 Mailing list traffic

Monthly mailing list traffic over the last 13 ½ years.

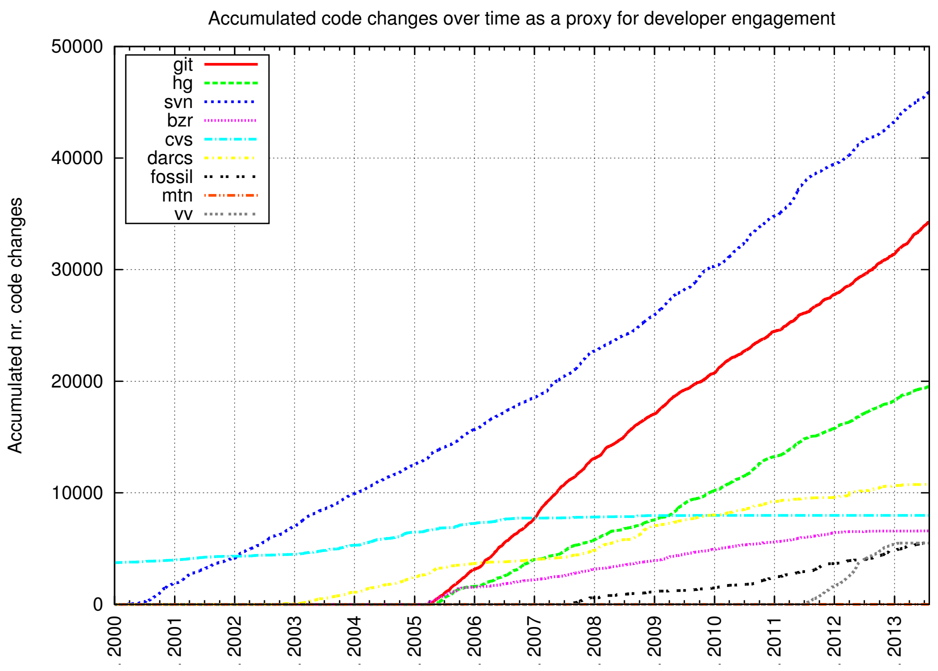


This metric indicates results similar to the Google trends analysis. SVN Q&A is rapidly declining and is being largely replaced by Git, and to a lower degree by Mercurial. CVS Q&A effectively died in 2008.

## 4 Developer Community engagement analysis

### 4.1 Code activity

Accumulated changes to the core SCM system over time (one code change is equivalent to one multifile task or commit).

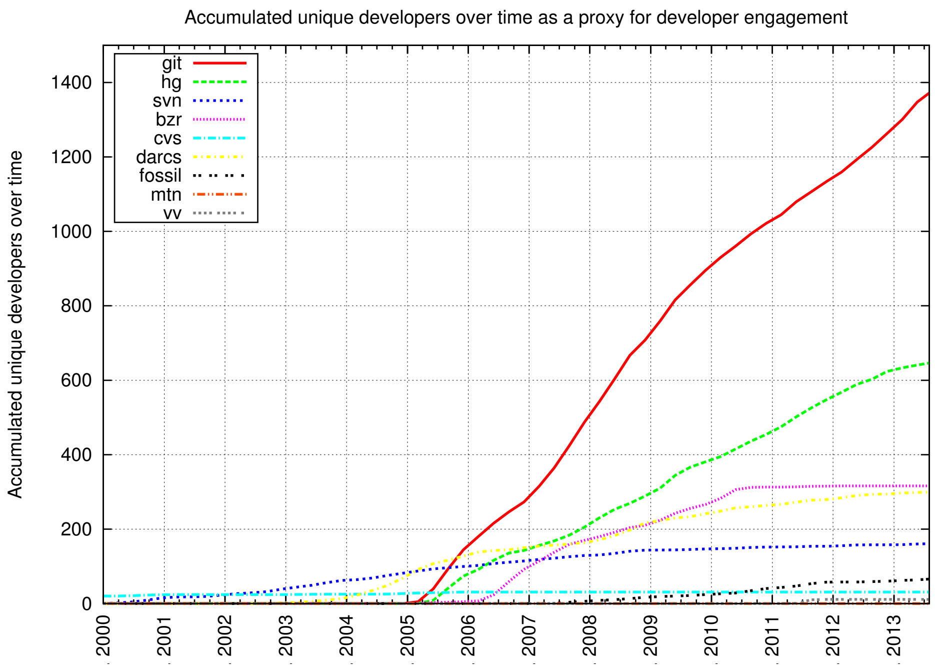


An interesting trend when looking at the key DVCS systems (Git and Mercurial), is the curvature of the graph. I.e. even though Git and Mercurial development started at the same time, almost twice as many code changes (requirements and bug fixes) have been incorporated into Git, the trajectory of the lines would indicate that this trend will continue for the foreseeable future.

SVN has a head start over the distributed systems when it comes to code changes over time; but due to it's "single file" CVS/RCS-inspired nature there is still today a tendency among the core developers to create a new change every time a single file is changed, somewhat inflating numbers as compared to the other systems.

## 4.2 Developer renewal and diversity

Accumulated unique developers that have contributed to developing the core SCM system over time.



Like the previous graphic this one also illustrates the ~2x ratio between developer engagement in git and mercurial, indicating higher innovation rate and significantly shorter requirement and issue turnaround time for Git.

It is also clear from the above graphic that some development communities are better able to integrate changes from new contributors than others, which could indicate better architecture and certainly broader developer mindshare.

## 4.3 COCOMO cost modelling

Estimated man years and cost to recreate based on COCOMO cost modelling, using current average US developer salary data.

| SCM System | Lines of code | Man Years | Development Cost (USD) |
|---|---|---|---|
| | | | |
| git | 327,740 | 87.57 | 19,124,432 |
| mercurial | 76,113 | 18.90 | 4,128,710 |
| bazaar | 231,550 | 60.80 | 13,278,825 |
| cvs | 174,327 | 45.13 | 9,856,336 |
| darcs | 47,234 | 11.46 | 2,501,786 |
| fossil | 217,821 | 57.02 | 12,453,384 |
| monotone | 209,944 | 54.86 | 11,980,950 |
| svn | 553,385 | 151.78 | 33,148,309 |
| veracity | 420,189 | 113.67 | 24,825,588 |

### 4.4 Secondary developer engagement

Frontends, wrapper scripts and usability tools play an important role in the day to day use of any SCM system, here we tabulate the number of surrounding tools mentioned on the different SCM's web and wiki pages.

| SCM System | Surrounding tools | Source |
|---|---|---|
| Git | ~175 | https://git.wiki.kernel.org/index.php/Interfaces,_frontends,_and_tools |
| Mercurial (hg) | ~76 | http://mercurial.selenic.com/wiki/OtherTools |
| Subversion (svn) | ~50 | https://en.wikipedia.org/wiki/Comparison_of_Subversion_clients |
| Bazaar (bzr) | ~30 | http://wiki.bazaar.canonical.com/3rdPartyTools |
| CVS | ~20 | http://ximbiot.com/cvs/cvshome/dev/addons.html |
| Darcs | ~30 | http://darcs.net/RelatedSoftware |
| Fossil | - | - |
| Monotone (mtn) | ~10 | http://wiki.monotone.ca/InterfacesFrontendsAndTools/ |
| Veracity (vv) | - | - |

## 5 Conclusion

For maximum ROI select only tools with

- highest adoption
- highest mindshare
- largest active user community
- largest active developer community
- highest investment to date
- bright and predictable future

Looking at the above graphs and numbers it is clear that only Git is currently worth granting serious consideration.

SVN can be easily excluded as it applies the legacy centralized data storage model, which is no longer considered technically current or relevant within the SCM field. Mercurial can be excluded based on very low Google trends search volume, low and declining mailing list traffic, a significantly lower accumulated number of developers over time, which taken together all indicate lower innovation and development rate. Basically the field quickly narrows to only *one* sensible choice, namely Git.

For further (more classical feature comparison based) reasons for adopting Git, please refer to one of the many online system comparisons, such as http://thkoch2001.github.com/whygitisbetter/

### 5.1 Business recommendations

Switch-Gears makes the following SCM system recommendations.

- If you are not using an SCM system today, you should adopt Git as a matter of course.
- If you are currently using a proprietary, homemade, or lower-ranked open source SCM system, you should migrate to Git as soon as possible.
- If you are currently implementing SCM in a heterogeneous manner, using a combination of various SCM systems, you should standardise on Git as soon as possible.