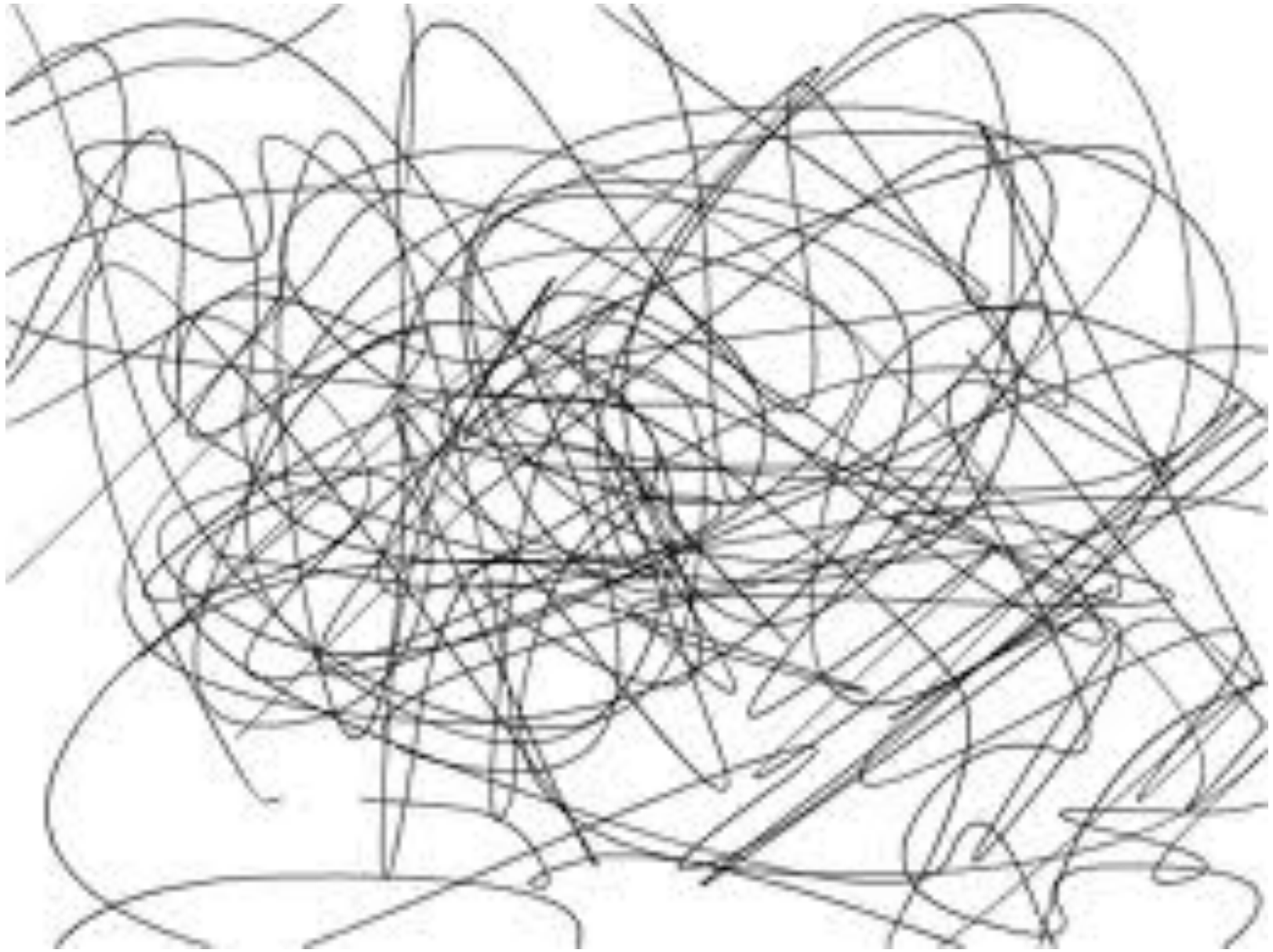
Exam Review

Peter C Rigby

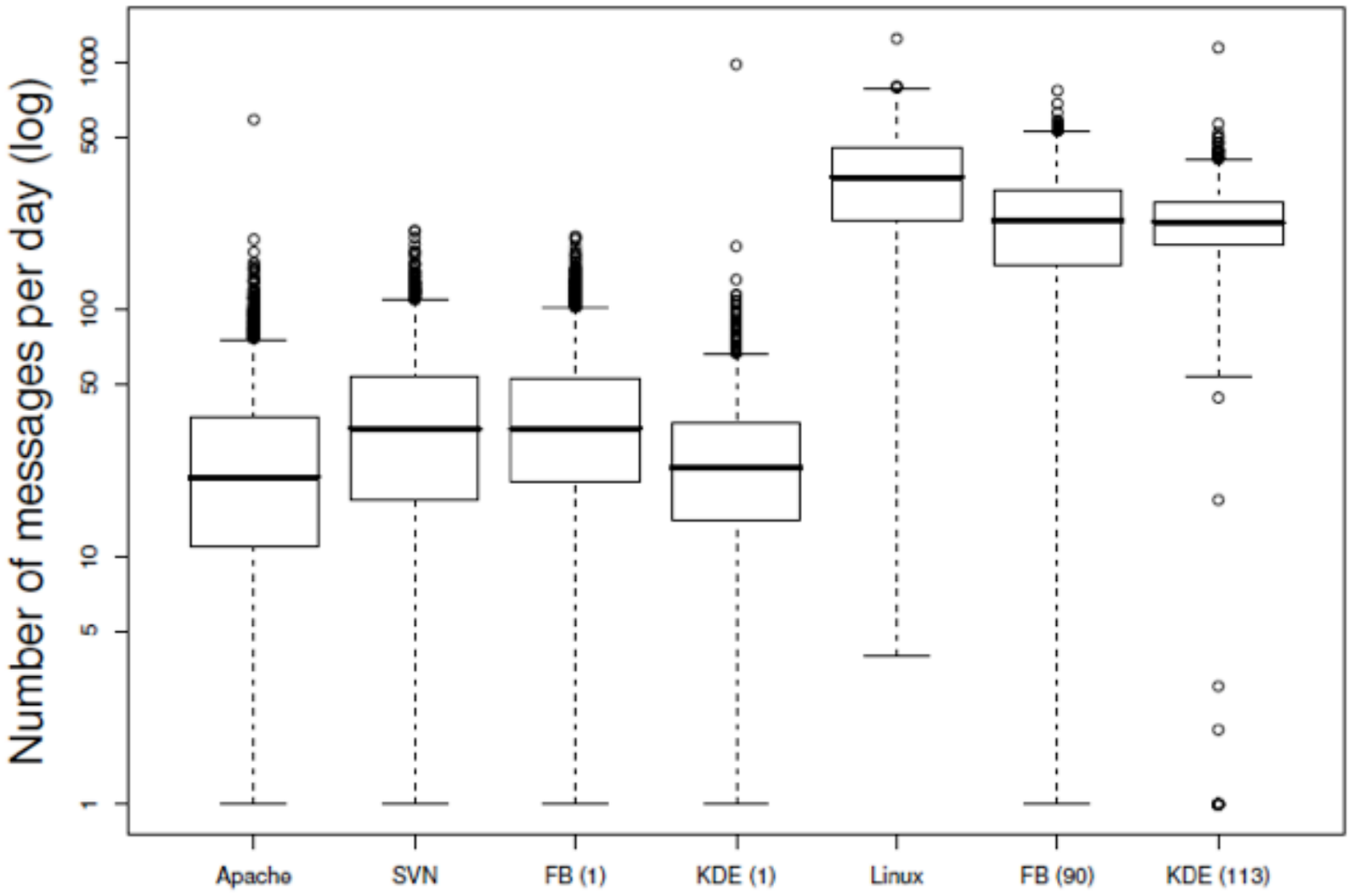
The Myths of Open Source Software (Introduction)

Peter C Rigby

Hundreds of Contributors?



Chaos



Massive Communication


VOLUNTEERS

Hobbyists?

No process?

Readings

Basic Measures (PotentialProjects)

Peter C Rigby

Paper Structure

- Abstract
- Introduction
- Background and Literature
- Methodology and Data
- Results
- Discussion
- Conclusion

Correlation

- The strength of the relationship between measures
  - e.g., demand for a product and its price
- Between -1 and 1
  - Strong .7 - 1.0
  - Moderate .5 - .7
  - Weak .3 - .5
  - Very weak .1 - .3
- What does a negative correlation mean?

Corrleation between Complexity and Size

	SLOC	LOC	FUNC
SLOC	1		
LOC	9844	1	
FUNC	8216	8187	1
BLKL	9186	9364	8203
CMTL	6742	7676	5477
CMTN	7612	7991	6018
CYCLO	9325	9164	7856
RETUR	7496	7469	7498
HLENG	9817	9655	7942
HVOLU	9815	9652	7921
HLEVE	9033	8851	7005
HMD	9715	8882	7737

Parsimony

- Parsimony is the scientific idea that the simplest explanation of a phenomenon is the best one.

Bikeshed

- Divide devs into core and non-core
- What is the impact of non-core developers
- Distracting or open for new knowledge?
- Rigby2011ICSE

Code Ownership and Code Quality

- "Don't Touch My Code! Examining the Effects of Ownership on Software Quality"
- Bird2011FSE

Intellectual Property

Peter C Rigby

4 types of IP

- Trade secret. The protection exists as long as the IP is kept secret.
- Copyright. It protects the expression of an idea for a limited period.
- Patent. It protects an invention, monopolistic exploitation of the invention for a period of time.
- Trademark. It protects a name, sound, image, etc. as associated with a particular type of business.

- Be able to define each in the context of software development.

Copyright

- It gives to the owner of the copyright of a work the **exclusive** right to:
  - 1. reproduce the work in copies;
  - 2. to prepare derivative works based upon the work;
  - 3. to distribute copies of the work to the public by sale or other transfer of ownership, or by rental, lease, or lending;
  - 4. to perform the work publicly;
  - 5. and to display the work publicly.
- Only the owner of the copyright has the right to sue to enforce his/her/its rights

Patents

- A patent applicant discloses the invention to the public
- In exchange the owner gains a monopoly over such invention for the duration of the patent (usually 20 years)
- Patents have national jurisdiction.
- Some countries, primarily the USA, issue software patents.

Patent trolls

- Patent holding companies whose only business is to make money from such patents
- Usually wait until their patents are infringed
- Example: NTP Inc:
  - Owns approximately 40 patents
  - RIM payed $612.5 to NTP Inc. to settle a long patent lawsuit (2006)
  - Then sued AT&T, Sprint Nextel, T-Mobile and Verizon Wireless.

Protecting yourself

- In theory one should verify if a new program/product might infringe a patent:
- Doing patent searches
  - Problematic: when patents are filled, they are secret,but they become valid from the day they are filled
  - Often difficult to determine if a patent is used in a product
  - Willful infringement a risk
- In practice it is always a risk

Collective works

- A collective work is “a work ... in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole” (17 USC)

Derivative Work

- A derivative work is "a work based upon one or more preexisting works, such as a translation.... or any other form in which work may be recast, transformed or adapted" (17 USC)

License

- A license is the legal document that grants a permission.
- A **software license** is the legal mechanism used by the software's copyright owner to grant certain permissions andrestrictions to the user.
- A **bare license**: A grant by the holder of a copyright or patent to another of any of the rights embodied in the copyright or patent, short of an assignment of all rights [Merriam-Webster's Dictionary of Law, 1996]
- According to [Rosen 04] "it is possible to grant a license to copy, modify, and distribute software without signing a contract between the two parties."

Open Source Software and Licenses

Peter C Rigby

Open Source License Definition
Know the first four

- 1. Free redistribution
- 2. Source code
- 3. Derived (derivative) works
- 4. Integrity of the author's source code
- 5. No discrimination against persons or groups
- 6. No discrimination against fields of endeavours
- 7. Redistribution of the license
- 8. License must not be specific to a product
- 9. License must not restrict other software
- 10. License must be technology-neutral

Types of Open Source Licenses [Ros04]

- Academic or Permissive: they allow the licensee a wide range of rights, including creating proprietary derivative works: BSD, MIT, Apache
- Reciprocal: they give you rights, but in return you should release any derivative works under the same license: GPL
- Links to software are permissible, but changes to core software are reciprocal (LGPL, MPL)

Reciprocal or Copyleft

- “Copyleft (a play on the word copyright) is the practice of using copyright law to offer the right to distribute copies and modified versions of a work and requiring that the same rights be preserved in modified versions of the work.”, wikipedia

Linking

- "As there is no record of anyone circumventing the GPL by dynamic linking without backing down when threatened with lawsuits by the copyright holder, the restriction is apparently de facto enforceable even if not currently de jure"
- [Wikipedia]

Compatibility of Licenses

- OSS licenses might impose contradictory requirements.
- When it is impossible to satisfy the requirements of two different licenses we say these two licenses are incompatible with each other.

- For example:
  - GPL: You cannot impose any further restrictions on this license to redistribute the software.
  - Old-BSD: You must advertise in all your materials the inclusion of the licensed software
- In such cases the two licenses are said to be non-compatible

References

- [Fre07] Free Software Foundation. FSF Web Site. http://fsf.org, 2007.
- [Ope07] Open Source Initiative. OSI Web Site. http://opensource.org, 2007.
- [Ros04] Lawrence Rosen. Open Source Licensing: Software Freedom and Intellectual Property Law. Prentice Hall, 2004.
- [Whe07] David Wheeler. Make your open source software gpl-compatible. or else. http://www.dwheeler.com/essays/gpl-compatible.html, 2007.

Introduction to OSS Business Models

Peter C Rigby

Business Models

- Standard
  - Make it Proprietary
  - Functional Encapsulation
  - Support Services and Training
- Dual-license Model
  - Proprietary Components
  - Freemum
  - Delayed Open Sourcing

Business Models

- Tricky
  - Software as a Service (SaaS)
  - Advertising-Supported Software
- Other factors
  - Branding
  - Donations
- Infrastructure
  - Paid Developers
  - Proprietary Modules

Business Models

- Be able to
  - Define/motivation
  - Risks
  - Example of Model

Hardware

- Charge for the hardware
  - Includes only the running binaries
- Release the source code
  - Usually don’t release the binaries

Hardware

- Risk
  - If your proprietary hardware is similar to a competitor,
  - they may be able to use your source code, and
  - release a similar product

Reference

- See links in document

- Wikipedia provides an good overview, their page is under revision as this is a evolving area

Version Control and Configuration Management (Branching)

Peter C Rigby

Terminology

- Commit: To store a change in the version control system in such a way that it can be incorporated into future releases of the project
- Log Message: A detailed description describing the change, can include reviewer information and links to bugs and other artifacts
- Repository: A software system that stores the change history
- Push/Commit: Send commit to another repository
- Update/Pull/Checkout: Obtaining a copy of the project from a repository

Isolation

- Good isolation
  - Developers work in parallel without making conflicting changes
- Bad isolation
  - Developers unknowingly make conflicting changes
  - Obvious conflicts (e.g., change the same line)
  - Violations of architecture or API (more serious bugs)

“Work of Named Stable Bases”

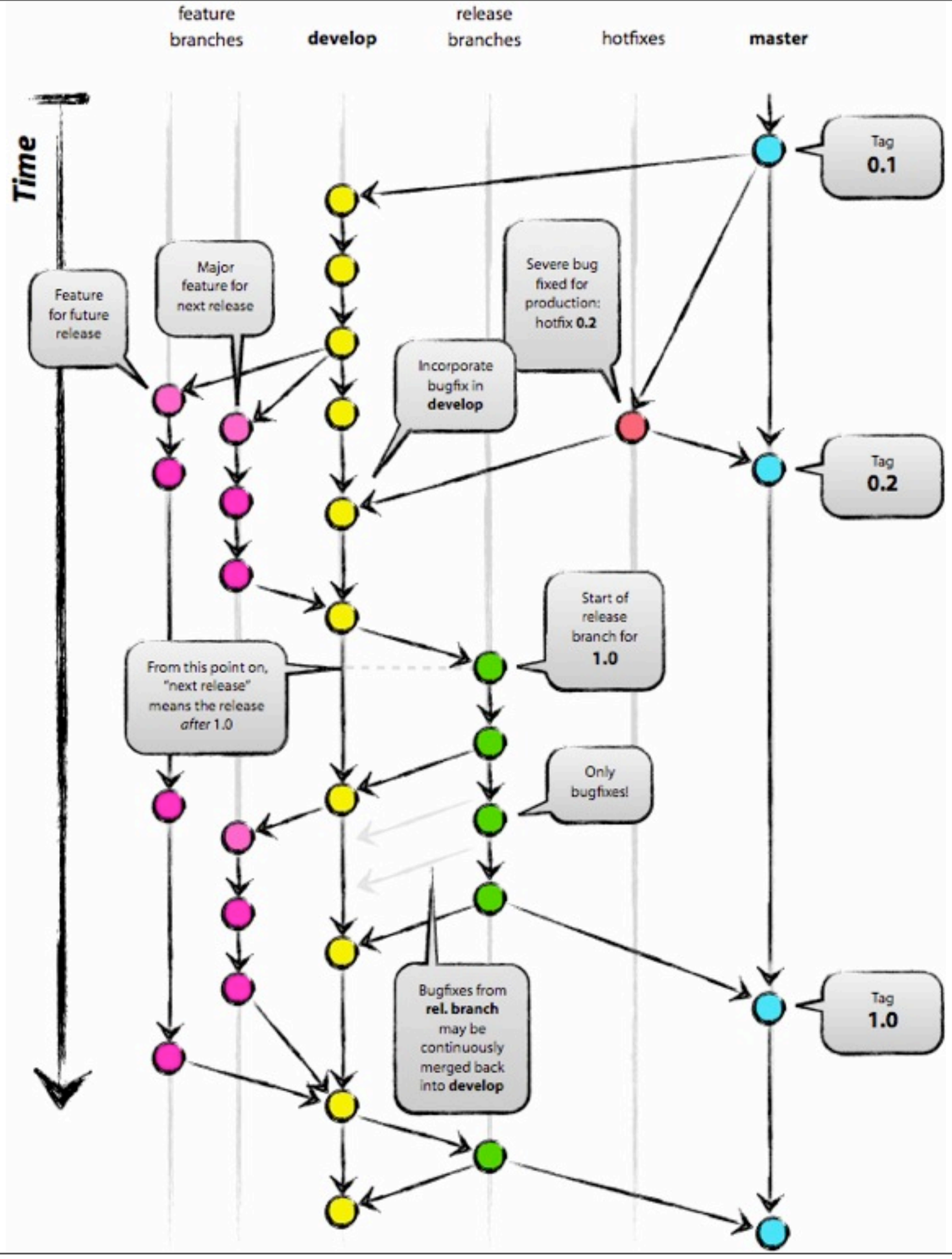
- Bases, tags, releases are of a known stability, set features and level of testing (no randomness!)
- Advantages
  - Developers are isolated from unrelated changes while they work
  - If something breaks, you broke it
  - Deal with familiar code (sandboxes)
- Disadvantage
  - The longer you wait the more likely you are to end up in “merge hell”, so “merge early, merge often”
  - Merges are usually done more frequently between subgroups

“Work of Named Stable Bases”

- Bases, tags, releases are of a known stability, set features and level of testing (no randomness!)
- Advantages
  - Developers are isolated from unrelated changes while they work
  - If something breaks, you broke it
  - Deal with familiar code (sandboxes)
- Disadvantage
  - The longer you wait the more likely you are to end up in “merge hell”, so “merge early, merge often”
  - Merges are usually done more frequently between subgroups

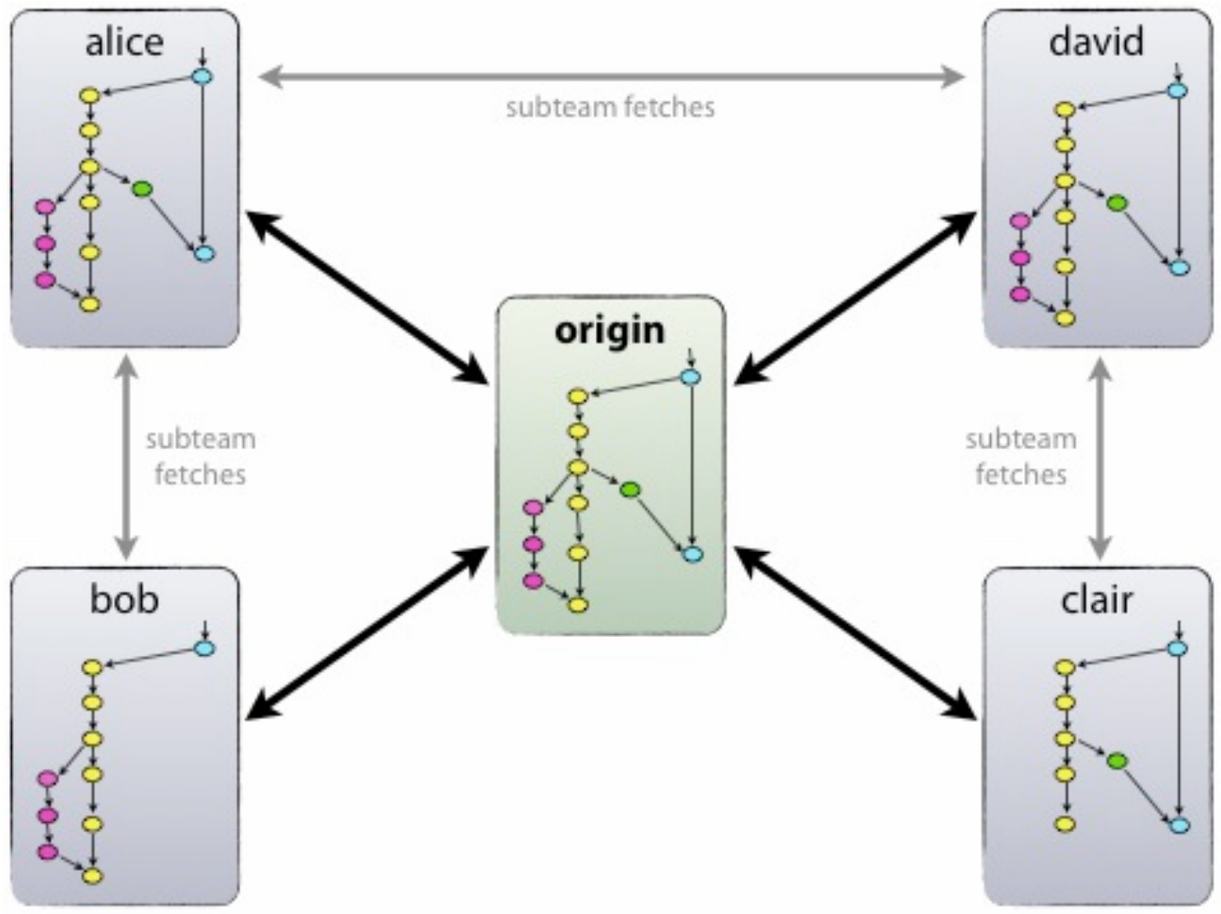
Branching And Process

From:
http://nvie.com/archives/323



Mixing Centralized and Decentralized

- Central repository for integration and final changes (origin)
- Subteams
  - Sandboxes
  - Peer sharing



No repo is inherently more important, but some are socially more important

Reading a Unified Diff

- Most common format
- --- /path/to/original_file
- +++ /path/to/new_file
- Has three lines of context
- New lines and remove lines are interleaved
  - + new line
  - - remove lined
- @@ -old start, old lines +new start, new lines @@

Types of Releases

- Analogy (road maintenance)
- Shut down highway
  - –Convenient for workers
  - –All branches frozen
- Shut down one lane
  - –Convenient for others
  - –Release branch frozen, others are open
- How does distributed VC affect the process?

Release Numbering

- Major . Minor . Micro
  - –2.6.34
  - –3.11
- Major = version = series
- Minor = new releases of this series
- Micro = security fixes etc
- Others?
  - –Backport number

Major, Minor, Micro

- Micro: same minor series
  - Forward- and backward-compatible
  - Small changes: bug fixes only or very small enhancements to existing features
  - No new features
- Minor number: same major series
  - Backward-compatible,
  - Some new features
- Major number: new series
  - Not backwards or forwards compatible
  - New features or set of features

Readings

- Chapter 7 of *Producing OSS*
- *Branch model*
  - http://nvie.com/posts/a-successful-git-branching-model/

Git
An Architectural Case Study

Software Architecture

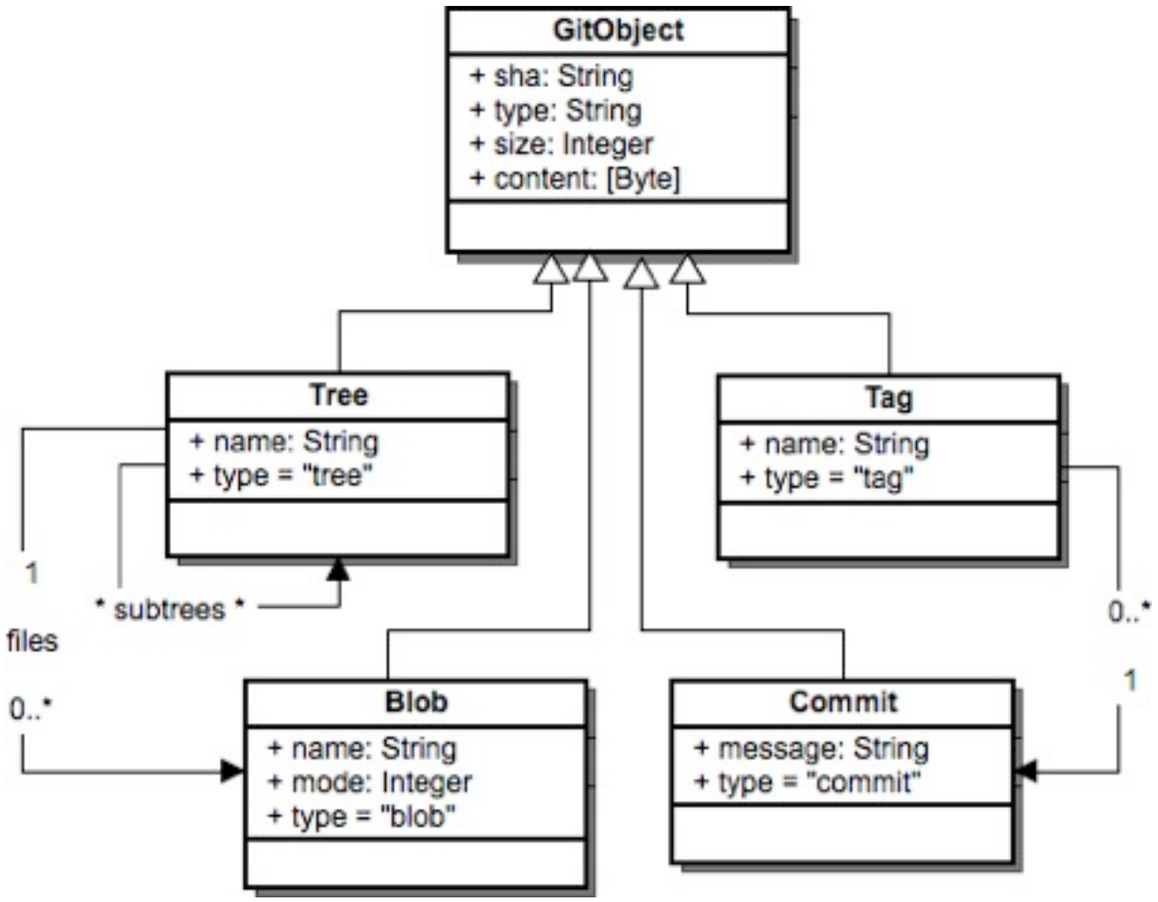
Peter C Rigby

The Architecture of Open Source Applications: Git -- Volume 2, Chapter 6
http://www.aosabook.org/en/git.html

Design Goals for git

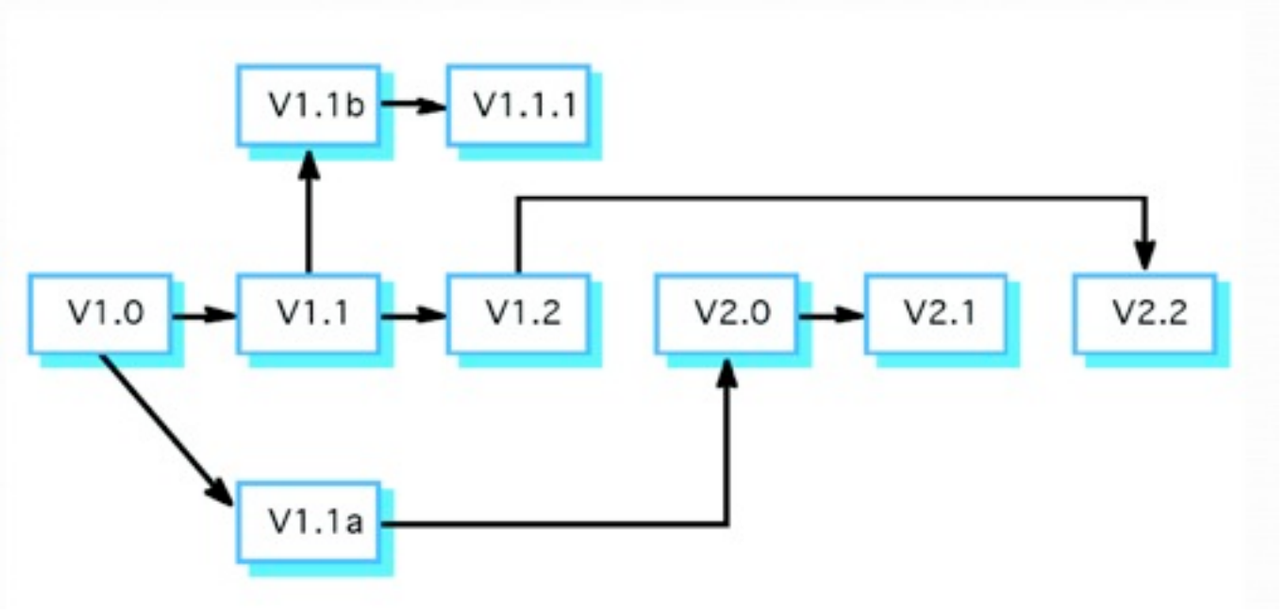
- Speed
- Simple design
- Fully distributed
- Strong support for non-linear development (thousands of parallel branches)
- Able to handle large projects like the Linux kernel efficiently (speed and data size)
- Safeguard against file corruption

Git Architecture



Hashing and DAG

- Each commit is uniquely identified by it's content
  - Branches do not need to be made a priori!
  - Can merge with anyone who has a common ancestor
- Directed Acyclic Graph
  - Multiple parents and children



Comparing Development Processes

Peter C Rigby

Development Processes

- Be able to contrast the different processes
- Code and fix
- Waterfall
- Spiral Model
- Agile
- RUP
- The Eclipse Way
- OSS and Linux

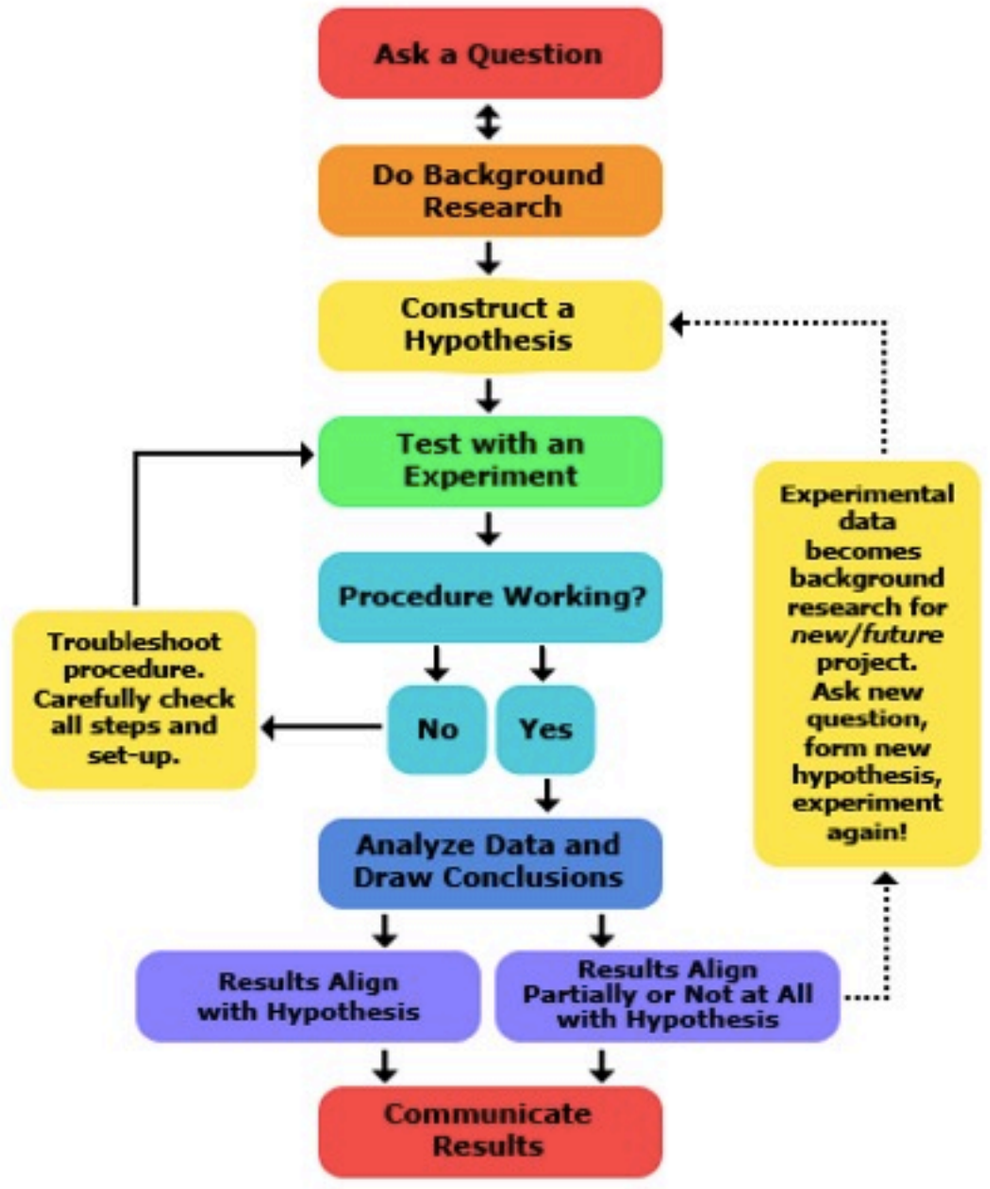
Research Methodologies

Peter C Rigby

Overview of Research Methodologies

- Qualitative Research
  - Ethnography, Case Study, Grounded Theory, Autobiography, Participatory Action Research, Phenomenology (each grounded in a specific discipline and philosophical assumptions)
- Quantitative Research
  - Case studies, survey methods, experiments
- Mixed Methods
  - Draw from qualitative and quantitative methods

Scientific Method



http://www.sciencebuddies.org/science-fair-projects/project_scientific_method.shtml#overviewofthescientificmethod

Good Hypotheses

- It is theoretically **grounded**: it is based upon literature relevant to the topic.
- It specifies the **relationship** between the values of two or more variables.
  - Direction (or **tendency**) of the relationship
- It makes a **testable** comparison using empirical data.

http://writing2.richmond.edu/writing/wweb/polisci/hypothesis.html

Confounds

Confounds

- The number of changes to a file is positively related to the number of bugs
- Confounds
  - –Expertise of developers working on the file
  - –Complexity of the file
  - –etc

Experiment

- Controlled environment
  - –All developers have similar training
  - –All work on the same files
- Vary only
  - –The number of changes developers make to the files
  - –How many bugs are introduced

Case Study

- “Phenomenon and context are not readily separable”, Yin
- The interactions are the most important part
- Setting is important
- Extract measures and model from real data

References

- Creswell, J.W. (2003). *Research design. Qualitative, quantitative and mixed methods approaches*
- R.K. Yin *Case Study Research: Design and Methods*
- Required reading:
  - Easterbrook et al *Selecting Empirical Methods for Software Engineering Research*
    - *http://www.cs.toronto.edu/~sme/papers/2007/SelectingEmpiricalMethods.pdf*