Version Control Studies
From Software Evolution to Mining Software Repositories

Thomas LaToza

What data do they use?
Do they follow good research methodology?
How do they help?

Why were they done?

How have they matured?

What problem are version control studies solving?

• Or in other words….
  – Couldn’t I have made my life much, much easier by picking a topic that has a review paper and / or is clearly scoped….

Researchers studying software evolution use version control studies

• Successful software changes
  – Users want new features
  – Fix bugs
  – Software has to work with different systems
• How does software react to changing requirements?
• Why does it get more difficult to change?

Version control studies are used by different people for different things

• Software evolution: 70s -
  – How does software adapt to changing requirements?
• Metrics: 80s -
  – What measures best predict the type or number of bugs?
• Empirical software engineering: 00s -
  – What are developers doing during a modification task?
• Programmer productivity: 00s -
  – What analyses of version control information are helpful for modifying code?
• Reverse of abstraction trends from functions to architecture
  – From “programming in the large” (decay of system over years) to “programming in the small” (provide useful information to developer for moment to moment tasks)
Maturation follows conference formation and bounded by software evolution and MSR

1975 International Conference on Software Engineering

1976 A Model of Large Program Development (Belady & Lehman)
  Code decays, causing slower nonlocal changes as design violated
  Redwine & Riddle – Concept formulation – how to deal with evolution

1976 International Conference on Software Maintenance
  Most software development cost is maintenance, how do we make it easier?

1984 [International] Conference on Software Maintenance

1984 [International] Conference on Software Maintenance

1998 International Workshop on Principles of Software Evolution (FSE workshop)

2004 Workshop on Mining Software Repositories (ICSE workshop)

Redwine & Riddle – Development and extension – how to do version control studies

Yikes, there's a lot of interesting stuff in version control repositories.

Redwine & Riddle – Development and extension – how to do version control studies

Neither software evolution studies nor modern version control studies have direct impact

- Software evolution
  - How does software change?
  - Most development work is change, change gets more expensive as code decays
  - Happened to use version control data to investigate
  - How do we make change cheaper?
  - A central motivation for many areas
  - Up front architecture, information hiding, program transformation, program families, software environments, refactoring, programming language features, software aging, agile software development, metrics, ...
  - Highly influential problem formulation, but studies of phenomena not at all influential

- Version control studies
  - Not first class until last 5 years with explosion of interest
  - No consideration of what is appropriate research methodology, data, questions, etc.
  - Just findings which are replicated and that few care about.....

But mature in that some phenomena have been studied many times

- Changes spread out over time [Lehman+ 76] [Belady+ 76] [Lawrence 82] [Yuen 87] [Eick+ 2001] [Purushothaman+ 2002] [Champaign 2003]
  - Talk about it differently – law of increasing entropy, code decay, types of dependencies, ripple effect
  - Little coherence
- Differences between adaptive, corrective, perfective maintenance [Kemerer+ 97]

Version control studies have very different mechanisms with which to benefit developers

- Managers – what should managers tell developers to do? [Belady+76] [Lehman+ 76]
  - Allocate perfective maintenance in most helpful places
  - Managers can better budget time for software modification
  - Used by business and process people
  - Developers should pay attention to defect prone changes and modules
  - Led to metrics suites to make it easier for developers to measure
- Tool evaluation – is this tool helpful? [Atkins+ 2002]
  - Tool developers can assess how their tool is useful and how it should be improved
  - Supplement to user studies, contextual inquiries, etc.
Version control studies have very different mechanisms with which to benefit developers

  - Developers should switch to more well-defined process and coordination mechanisms when project grows beyond 15 people
- Design principles – does removing code clones help? (Kim+ 2005)
  - Developers can better evaluate if they should spend effort removing code clones
  - Tool researchers can devise tools more effective at synchronizing code clones (linked editing)
- Tools – recommend maximally relevant artifact based on past changes (Ying+ 2004)
  - Tool provides developer artifact that is likely to be needed to be modified so they don’t forget to change it.
- Communities tied together by some people, otherwise don’t interact
  - Dewayne Perry worked with software evolution, Lucent version control studies, mining software repositories
  - Contributor to why version control studies immature

The space of version control data

- Modification request logs
  - Data about what they are doing
  - Time
  - Reason
  - Developer identity
- Checkins - version control system
  - # size components (functions, classes, files, directories, packages, …)
  - # type of connectors (call dependencies, …)
  - Location of changed lines
  - Developer identity
  - Time & date
- Bug logs
  - Cause
  - Where fix is localized
  - Time to fix
  - Developer identity
- Other
  - Programming language
  - Type of application

How has the important data used in a version control study changed?

- Software evolution
  - # modules, # modules modified by change, release frequency
  - Population - OS/360 (Belady+ 76)
- Defect prediction
  - Detailed cause, # modules modified by change, change time
  - Population - Maryland software engineering lab – embedded systems (Basili+ 84); SESS telephone switch (Perry+ 85)
- Tool assessment
  - Change time
  - Population - SESS telephone switch (Atkins+ 2002)
- Artifact recommendation
  - Modules modified by change to module
  - Population – what system is run on (Mozilla and Eclipse in paper) (Ying+ 2004)

How has the quality of research methodology matured?

- Version control studies are case studies of particular system(s)
  - Have to argue that findings have external validity
  - Give context variables sufficient for replication
- Report population and selection criteria
  - Why were these changes, systems, modules, etc. used?
- Don’t oversimplify and lose context
  - Are there other reasons for phenomena or effects of idea being proposed?
- State hypothesis based on explanation
  - Why do you expect there to be a difference
• Not fair to look at all studies
  – Most quality variation still between workshop and journal papers
    • How much data
    • Interestingness of findings
    • Quality of arguments
  – So, sample quality representative papers....

How have version control studies matured methodologically?
• Do changes disperse? [Belady+ 76]
  – Have to argue that findings have external validity
    • Horrible - jumps from OS/360 case to a “law” of program evolution!!!
  – Give context variables sufficient for replication
    • Horrible – no information on # developers, development tools, software process, design methodology, ...
    • Might exist elsewhere...
  – Don’t oversimplify and lose context
    • Partial – could consider other explanations for laws more
  – Report population and selection criteria
    • Good - talks about releases and gives reasons for exclusion of a release from consideration
    • State hypothesis based on explanation
    • Partial – gives explanation for “laws” but not all
  • Also does not benefit from quality of methodology...

How have version control studies matured methodologically?
• What predicts bugs? [Basili+ 84]
  – Have to argue that findings have external validity
    • Partial – conservatively states that needs more projects for generality, but doesn’t argue for generalizability of particular project studied
  – Give context variables sufficient for replication
    • Horrible – no information on # developers, development tools, software process, design methodology, ...
  – Report population and selection criteria
    • Good – reports on which changes, includes everything
  – Don’t oversimplify and lose context
    • Partial – needs more alternative explanations
  – State hypothesis based on explanation
    • Partial – just exploratory description, no hypotheses tested

How have version control studies matured methodologically?
• Using version control studies to assess tool [Atkins+ 2002]
  – Have to argue that findings have external validity
    • Ok – acknowledges case study and tries to argue that other systems wouldn’t be that different
  – Give context variables sufficient for replication
    • Good – gives lots of context info
  – Report population and selection criteria
    • Good – talks about how changes selected and description of population
  – Don’t oversimplify and lose context
    • Partial - abstract claims “developers were 40% more productive”, adding on second page “(when changing files containing preprocessor directives)”
    • State hypothesis based on explanation
    • Partial – superficial hypothesis, but gives some “anecdotal” reports for explanation
Conclusions – Version Control

Studies have a Bright Future

• Promising area of research
  – Lots of easy to obtain data with rise of open source
    • No expense of conducting experiment…
    • But can’t manipulate anything…..
  – Theories, phenomena examined to data have been really limited
    • Recent interest in more general phenomena like code duplication or coordination
  – Promising application to tools
    • Artifact recommendation, duplication detectors
    • Defect detectors next?
    • But needs to be based on solid, predictive models rather than silly models…..

Questions?