Digital Archeology

Audris Mockus

audris@avaya.com

Avaya Labs Research
Basking Ridge, NJ 07920
http://mockus.org/

Outline

- Why Measure?
- What is Digital Archeology?
- Examples:
  - Impact of ownership transfer: discovery of mentoring relationships
  - Retention of contributors: measures of willingness and climate
- Synthesis
- Future

Why measure?

"... the art of *measurement* would do away with the effect of appearances, and, showing the *truth*, would fain teach the soul at last to find rest in the truth, and would thus save our life."

Protagoras, Plato

The absence of romance in my history will, I fear, detract somewhat from its interest; but if it be judged useful by those inquirers who desire an *exact knowledge of the past* as an aid to the interpretation of the future, which in the course of *human things* must *resemble* if it does not *reflect* it, I shall be content.

The History of the Peloponnesian War, Thucydides

Science(s) of human and collective nature

- A is the study of *past human events and activities*
- B is the study of human **cultures** through the *recovery, documentation and analysis of* **material** *remains*
- C is the study of developer **culture** and **behavior** through the *recovery, documentation and analysis of* **digital** *remains*

Digital Archeology

- The study of developer **culture** and **behavior** through the *recovery, documentation and analysis of* **digital** *remains*
- Primary method
  - *Organizational Tomography* is the reconstruction of people's behavior from the observed projections in digital remains
    - By linking traces from
      - unrelated tools, e.g., by using
        - Chronology of events
        - Patterns of individual and social behavior
        - Nature of roles and tasks
- Commitment
  - Understand human endeavor with increasing *precision* and *scope*

Why Start with Software Development?

Tools produce detailed “digital remains”

Software is essential for economy

Work, play, and commerce move to digital environments

Can easily adapt to study any human endeavor

Examples of Digital Remains

Code tools: Version Control Systems

(e.g., CVS/SVN/Git/Mercurial/Bazaar/ClearCase)

Task/Workflow tools: issue (MR) tracking systems

(e.g., Bugzilla, Jira, ClearQuest, Siebel)

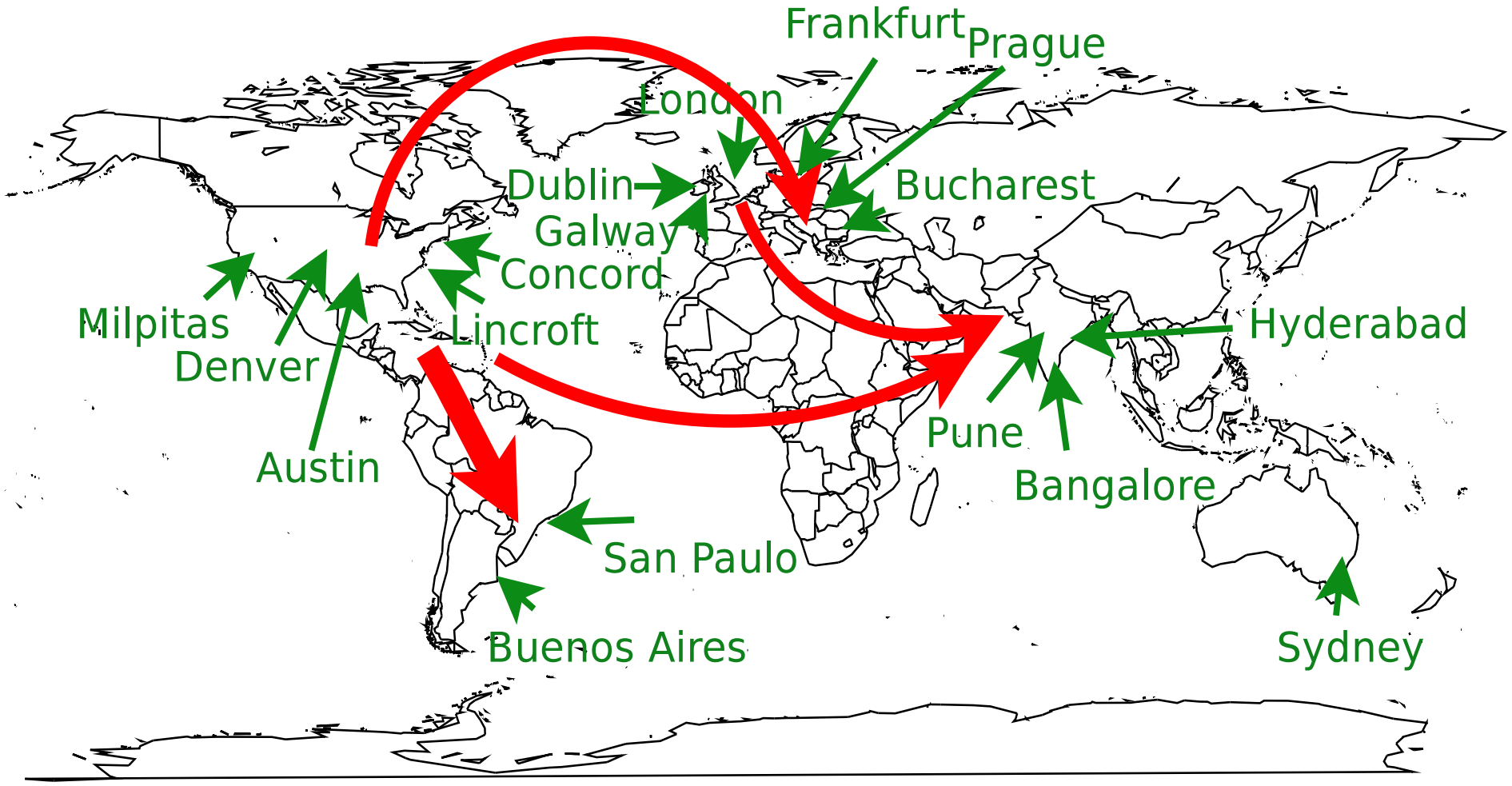
Other tools: Usage and communication, sales, org.

(e.g., mailing list, web log, SAP, PeopleSoft, ...)

Understand how humans work with increasing precision and *scope*

Scope	Research Questions
→ **Individual**	Expertise, productivity
Project	Skills, collaboration, learning
Community/Organization	Willingness, innovation
Society	(Truth, beauty), knowledge

Succession: Transfer of Ownership



17 Avaya development locations: ≈ 3K developers

Phenomena of Succession

- Research question: observe *succession* and quantify its impact
- Terminology for basic concepts
  - *Implicit teams* are groups based on the affinity to the parts of the product they work(ed) on
  - *Succession* is the transfer of responsibilities to maintain and enhance the product within an *implicit team*
  - *Follower* is the receiving party
  - *Mentor* is the transferring party

Traces of development: software changes

Before:

```
int i = n;
while(i++)
    prinf(" %d", i−−);
```

After:

```
//print n integers
int i = n;
while(i++ && i > 0)
    prinf(" %d", i−−);
```

- Change data
  - Date, login, defect number, . . .
  - 1K followers/13 products in ClearCase, SCCS, CVS, and SVN
- Basic cleaning
  - Map logins to people via NIS and POST dated snapshots
  - Exclude VCS administrators and automatic changes

Remains of Succession

- Key assumptions for developer work patterns
  - "Engaging" with the code often leads to changing the code
  - *Mentors* precede *followers* in the temporal order of changes
- Reconstruction or tomography
  - Implicit teams: developers changing the same files:

$$ImplicitTeam(d_i, d_j) \iff$$
$$\exists File, Touches(d_i, File) \wedge Touches(d_j, File)$$

  - Reconstructed mentor m for developer d: maximize succession signature (likelihood) S

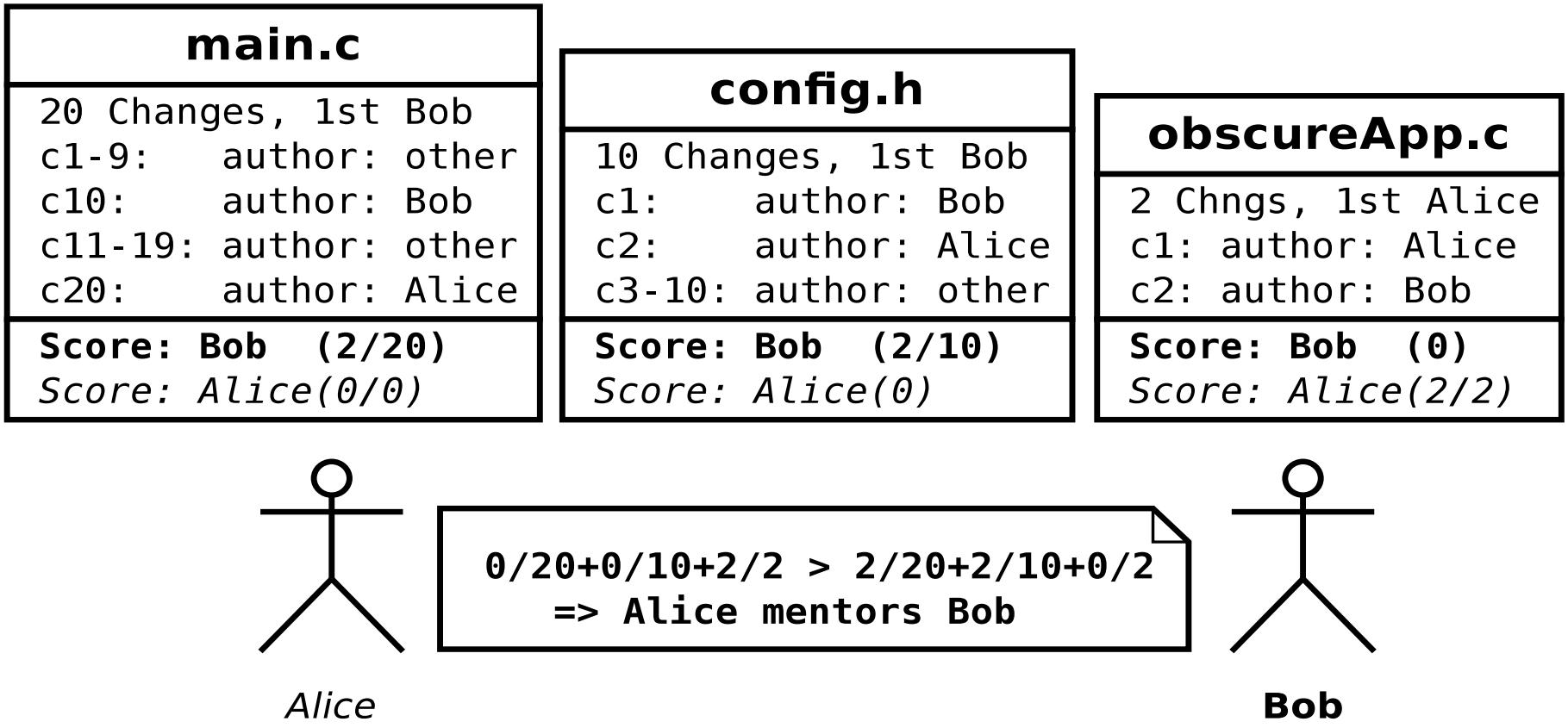
$$m = \arg \max_{M \in \{ImplicitTeam(d,M)\}} S(d, M)$$

  - Designed and evaluated four succession signatures $S_1, ..., S_4$

Best succession signature

Based on interview-derived ten mentor-follower pairs

Weight each file by the fraction of file's changes done by the pair



Impact of succession: productivity ratio (PR)

$$PR = \frac{Productivity(Follower)}{Productivity(Mentor)}$$

- Productivity: number of delta (atomic changes) per month
- Organizational Socialization theory to create the model for PR

$$\begin{aligned}\log(PR) \quad = \quad & Time + MentorOffshore + \\ & PrimaryArea + \\ & ExpertiseBreadth + \\ & ProjectSize + \log(Number\ of\ Followers).\end{aligned}$$

$\frac{\text{Follower productivity}}{\text{Mentor productivity}}$

	Estimate	p-value	Effect size
Time of transfer	-0.01	0.31	
Mentor Offshore	-0.63	0.00	**1/2**
Not primary expertise	-0.68	0.00	**1/2**
Mentor's breadth of expertise	-1.41	0.00	**1/2**
Large-scale products	-1.21	0.00	**1/3**
Medium-scale products	-0.46	0.00	**2/3**
$\ln(NF)$	-0.53	0.00	$\mathbf{1/\sqrt{NF}}$

Use S_2 to determine mentor for each of the 1012 followers

Adjusted $R^2 = 0.59$.

Practical implications

- Matches observed empirical rule (a team of four or five to replace one experienced developer)
- Implemented following recommendations
  - Start with small and new projects
  - Take more time to transfer
  - Do not overload mentors with too many followers
  - Transfer mentor's primary expertise

Results: Community Scope

Scope	Research Questions
Individual	Expertise, productivity
Project	Skills, learning
→ **Community/Organization**	Motivation, innovation
Society	(Truth, beauty), knowledge

Retention Hypothesis

- Motivation
  - It takes more than three years to become fluent in a large project
  - Many sites had median tenure of less than 1.5 years
  - Training and mentoring large numbers of newcomers drains resources
- Hypothesis
  - A contributor tenure with the project depends on
    - Individual's *willingness* and *capability*
    - The *environment* she encounters when joining
- Can long tenure (> 3 years) be predicted based on the digital traces left during the first month of participation?

Traces: issue workflow

- Life of an issue
  - Born, e.g., a user encounters and reports an issue $\rightarrow$ *unconfirmed*
  - Triaged: issue is inspected $\rightarrow$ *new* or *needinfo*
  - Comments added, e.g, suggestions, additional information
  - Resolved: e.g., autoclose, fix, nochange
- Data
  - Mozilla 700K, Gnome 600K issues (three sources), 2001-2011
  - Project pages and their history, published literature
  - Interviews/Surveys
- Cleaning/Augmenting: remove bugmasters, map email/login to person, exclude last three years, identify crash reporter, group outcomes

How to measure willingness?

- When barrier to reporting is low more low-willingness contributors join
  - Click a submit button in a crash-reporter
  - Open Bugzilla account and report the issue
- Direct indicators of willingness
  - Submit issue with the information needed to reproduce and fix it
  - Scan existing issues and a comment on the one that is most similar
- Possible measures
  - First action: breakpad, regular report, comment
  - Fraction of reports that are fixed

Prediction of long/productive tenure

		Odds Ratio		
Measure	Predictor	Mozilla	Gnome	Direction
Macro env.	# of Users	50%	25%	↓
	RelativeSociality	102%	114%	↑
Micro env	attention (lack of)	33%	66%	↓
	Min(peer experience)	139%	114%	↑
	Peer soc. clust.	168%	133%	↑
Willingness	%Fixed	131%	131%	↑
	FirstNotReport	600%	400%	↑
Capacity	# of Comments	119%	136%	↑
	# of peers	132%	114%	↑

Mozilla/Gnome (168723/124242 observations)

Willingness and opportunity affect tenure

For participants:

Take pro-community attitude

For projects:

Improve attention to newcomers

Examples of Organizational Tomography

- Measure a concept without **any** direct record
  - Change effort [3, 2]
  - Change purpose [9]
  - Mentorship [6]
  - Dimensions of expertise and fluency [8, 13]
  - Chunks (independently maintainable code) [11]
  - Measures of customer experience [5, 10, 12]
- Infer **unrecorded** relationships
  - A change and defects it causes [12]
  - Social history and quality [1]
  - Organizational change and quality [7]
  - Social dependencies and effort [4]
  - Relative sociality, climate, willingness and retention [14, 15]

Digital Archeology: Synthesis

Commitment: Understand human endeavor (e.g, how humans produce software) with increasing precision and scope

Paradigm:

- Select a phenomenon for study and amass observational data
- Observe and validate on a smaller scale how it projects onto digital remains.
- **Primary puzzle:** Design and validate models of the projection to reconstruct unobserved concepts
- Apply the suitable tomography method on the entire population to reconstruct the phenomenon and its impact
- Build higher-level concepts of human endeavor (software development) based on the validated lower-level reconstructions

Digital Remains

- Ongoing collection of Digital Remains
  - Avaya-wide
    - Code growth VCS(170MLOC)/Issue(13M)
    - Services (56M tickets),
    - Sales (>10M assets), Organization (20K people)
  - PKU Cloud
    - innovation spread (> 600K VCSes, 200M file/versions)
  - OSS services: environment and willingness (4M issues)
  - Wikipedias: compare culture (800 wikies, 200M edits)

Future challenges for Digital Archeology

- Increasing scope
  - Other types of human endeavor, e.g., services, games, entertainment
  - Individual activity at micro-scale
  - Collective activity at culture level
- Increasing precision
  - Linking individual and collective activities recorded at different scales
  - Models of context, e.g., environment and culture
  - Models of subconscious: e.g., passion, energy

Thank you

References

[1] Marcelo Cataldo, Audris Mockus, Jeffrey A. Roberts, and James D. Herbsleb. Software dependencies, the structure of work dependencies and their impact on failures. *IEEE Transactions on Software Engineering*, 2009.

[2] T. Graves and A. Mockus. Identifying productivity drivers by modeling work units using partial data. *Technometrics*, 43(2):168–179, May 2001.

[3] Todd L. Graves and Audris Mockus. Inferring change effort from configuration management data. In *Metrics 98: Fifth International Symposium on Software Metrics*, pages 267–273, Bethesda, Maryland, November 1998.

[4] James Herbsleb and Audris Mockus. Formulation and preliminary test of an empirical theory of coordination in software engineering. In *2003 International Conference on Foundations of Software Engineering*, Helsinki, Finland, October 2003. ACM Press.

[5] Audris Mockus. Empirical estimates of software availability of deployed systems. In *2006 International Symposium on Empirical Software Engineering*, pages 222–231, Rio de Janeiro, Brazil, September 21-22 2006. ACM Press.

[6] Audris Mockus. Succession: Measuring transfer of code and developer productivity. In *2009 International Conference on Software Engineering*,

Vancouver, CA, May 12–22 2009. ACM Press.

[7] Audris Mockus. Organizational volatility and its effects on software defects. In *ACM SIGSOFT / FSE*, pages 117–126, Santa Fe, New Mexico, November 7–11 2010.

[8] Audris Mockus and James Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In *2002 International Conference on Software Engineering*, pages 503–512, Orlando, Florida, May 19-25 2002. ACM Press.

[9] Audris Mockus and Lawrence G. Votta. Identifying reasons for software change using historic databases. In *International Conference on Software Maintenance*, pages 120–130, San Jose, California, October 11-14 2000.

[10] Audris Mockus and David Weiss. Interval quality: Relating customer-perceived quality to process quality. In *2008 International Conference on Software Engineering*, pages 733–740, Leipzig, Germany, May 10–18 2008. ACM Press.

[11] Audris Mockus and David M. Weiss. Globalization by chunking: a quantitative approach. *IEEE Software*, 18(2):30–37, March 2001.

[12] Audris Mockus, Ping Zhang, and Paul Li. Drivers for customer perceived software quality. In *ICSE 2005*, pages 225–233, St Louis, Missouri, May 2005.

ACM Press.

[13] Minghui Zhou and Audris Mockus. Developer fluency: Achieving true mastery in software projects. In *ACM SIGSOFT / FSE*, pages 137–146, Santa Fe, New Mexico, November 7–11 2010.

[14] Minghui Zhou and Audris Mockus. Does the initial environment impact the future of developers? In *ICSE 2011*, pages 271–280, Honolulu, Havaii, May 21–28 2011.

[15] Minghui Zhou and Audris Mockus. What make long term contributors: Willingness and opportunity in oss community. In *ICSE 2012*, page accepted, Zürich, Switzerland, 2012.

Abstract: Professional and social activities are increasingly software mediated thus generating vast digital traces representing projections of collective and individual actions. The reconstruction and quantification of the behavior of an individual, an organization, or a society from these projections is the main challenge of digital archeology. I will illustrate the approach using examples of radical changes in software development practices driven by the open source movement and the business needs to move development to low-cost locations. In particular, I will discuss the design end evaluation of the measures for mentor-follower relationships in code ownership. Success in open source and commercial projects critically depends on willing expert participants. Unfortunately, long-term active participation is necessary to acquire project's expertise. I will discuss the design of measures for the willingness of new contributors and for the climate they encounter when joining. More passionate joiners encountering a nurturing climate had dramatically greater chances to become long-term contributors in Mozilla and Gnome projects. I'll conclude by outlining approaches to data collection and by synthesizing the goals of digital archeology that may provide

unique insights into human nature.

Audris Mockus
Avaya Labs Research
233 Mt. Airy Road
Basking Ridge, NJ 07920
ph: +1 908 696 5608, fax:+1 908 696 5402
http://mockus.org, mailto:audris@mockus.org

Audris Mockus studies software developers' culture and behavior through the recovery, documentation, and analysis of digital remains. These digital traces reflect projections of collective and individual activity. He reconstructs the reality from these projections by designing data mining methods to summarize and augment these digital traces, interactive visualization techniques to inspect, present, and control the behavior of teams and individuals, and statistical models and optimization techniques to understand the nature of individual and collective behavior. Audris Mockus received B.S. and M.S. in Applied Mathematics from Moscow Institute of Physics and Technology in 1988. In 1991 he received M.S. and in 1994 he received Ph.D. in Statistics from Carnegie Mellon University. He works at Avaya Labs Research. Previously he worked at Software Production Research Department of Bell Labs.

A trivial reconstruction: Issue History

Group history records by issue and order by date

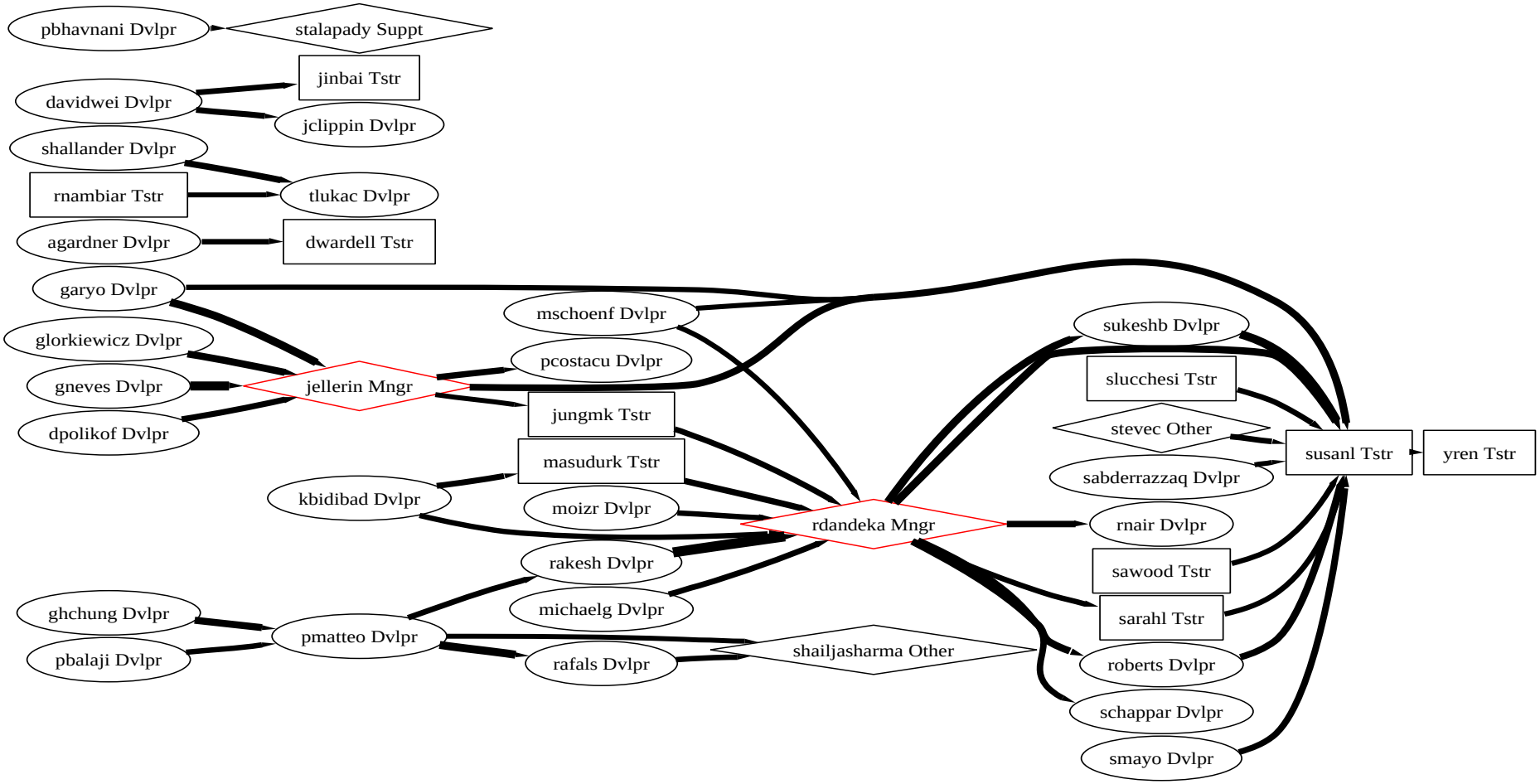
`bugzilla.gnome.org/show_activity.cgi?id=388441`

Who	When	What	Removed	Added
fherrera	2006-12-21	Status	UNCONFIRMED	ASSIGNE
chpe	2006-12-22	CC		chpe
fherrera	2006-12-22	Attachment #78764	0	1
herzi	2007-01-30	CC		herzi
chpe	2007-06-14	Blocks		436832
fherrera	2007-08-28	Status	ASSIGNED	RESOLVE
mbarnes	2008-06-09	CC		mbarnes

A simple reconstruction: workflow

Developer graph: A is linked to B with weight $W > const$

$$W = \|(Issue, i) : \exists(Issue, B, t_i) \wedge \exists(Issue, A, t_{i+1})\|$$



Results: Project

Why followers are so “unproductive”

lets look at the **project** level: How long it takes to become competent?

Scope	Research Questions
Individual	Expertise, productivity
→ **Project**	Skills, collaboration, learning
Community/Organization	Motivation, innovation
Society	(Truth, beauty), knowledge

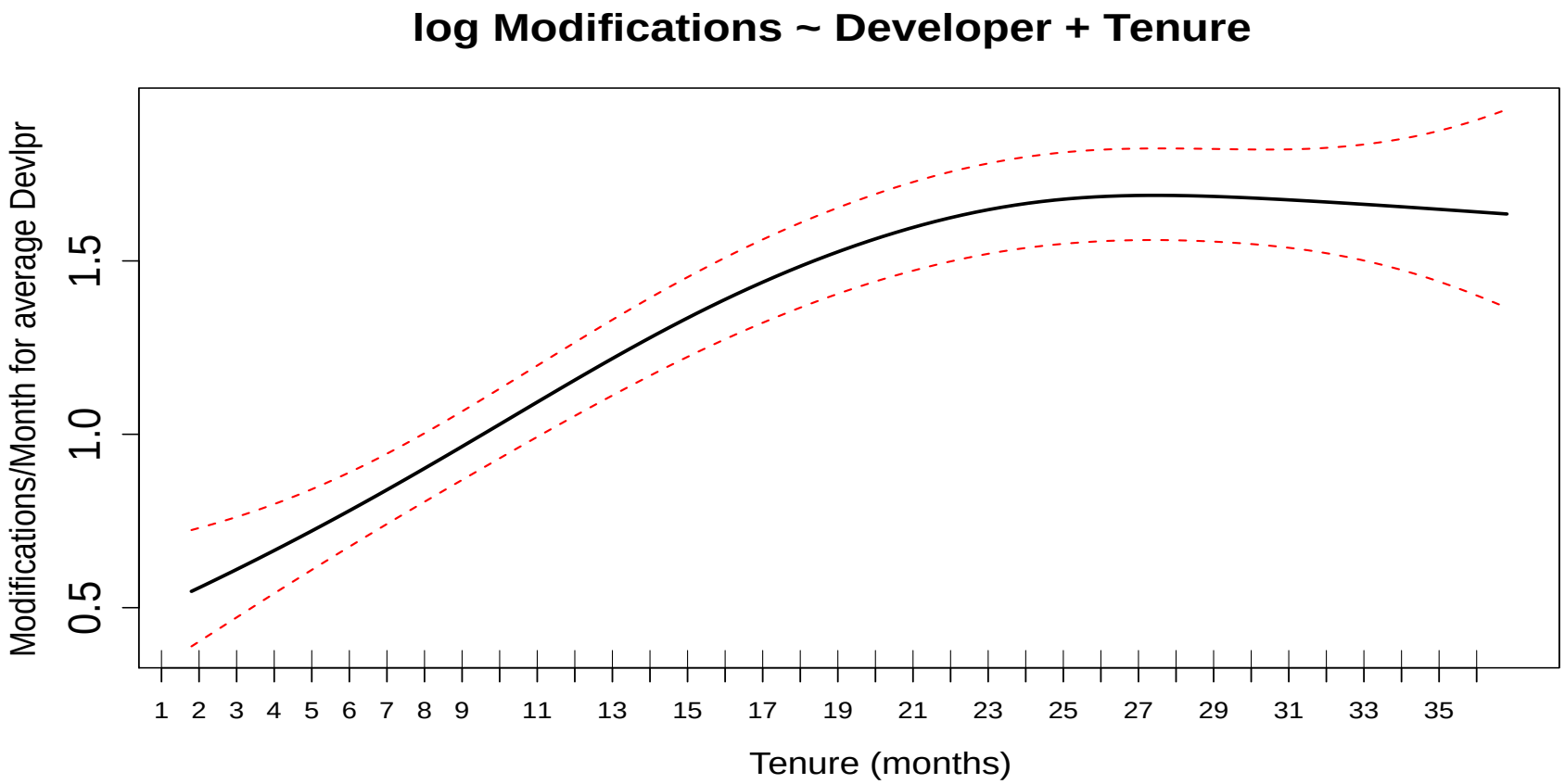
Paradox: Productive $\neq$ Competent

- How long does it take for a developer in your project to become productive?
  - Small-medium scale projects: 2-6 months
  - Large scale project: 12 months
- What are the stages for a developer?
  - Small-medium scale projects : “it takes several years to become competent in important tasks”
  - Large scale project : “we had attempted to assign mentoring tasks to developers with only two years of experience, but had unsatisfactory results”

How long to become fluent?

- **Competency**: the quality of being adequately or well qualified
- **Productivity**: the quantity of tasks completed per unit time
- **Fluency**: ability to complete project tasks rapidly and accurately independent of task difficulty or importance.

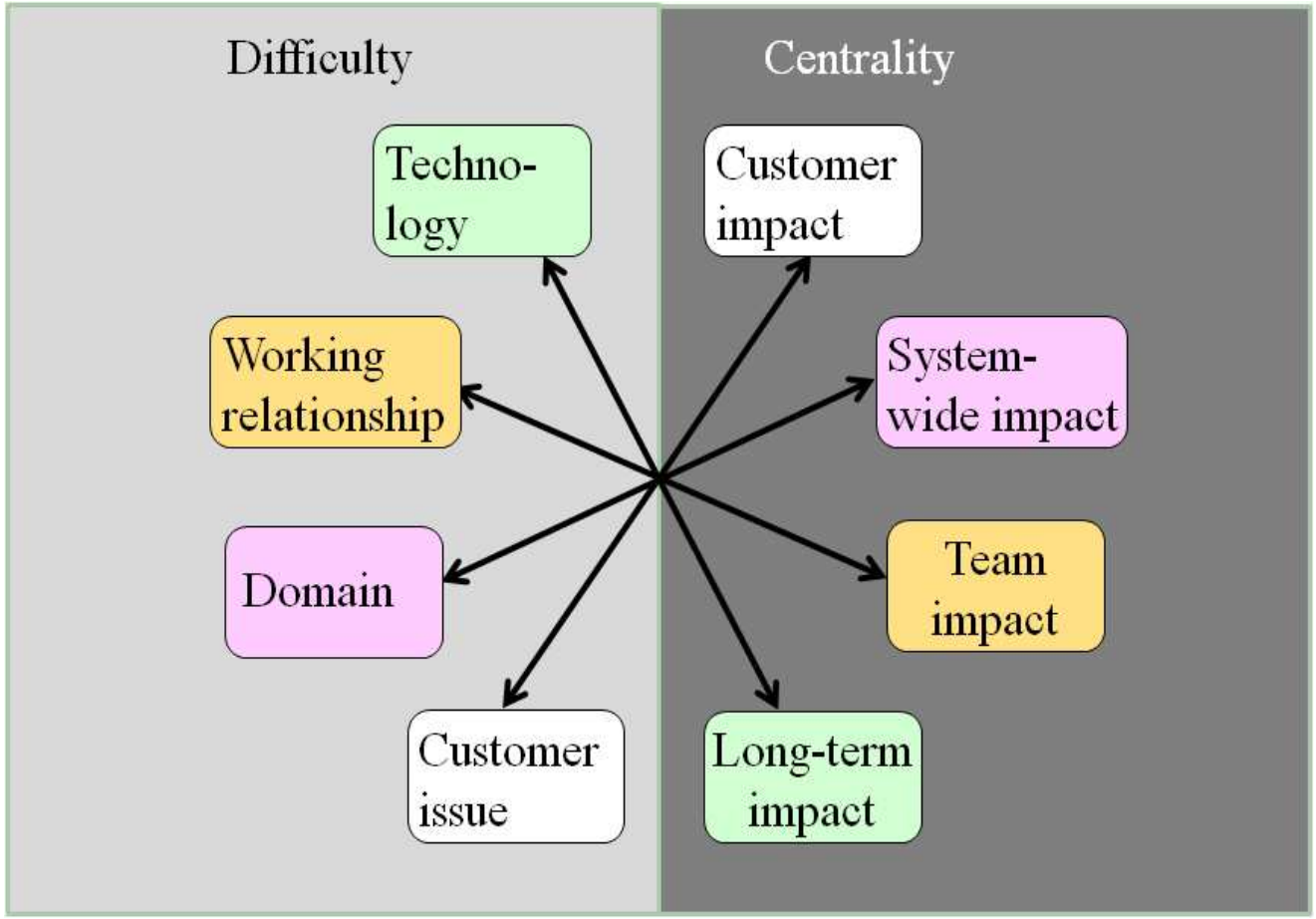
20 months in a large project



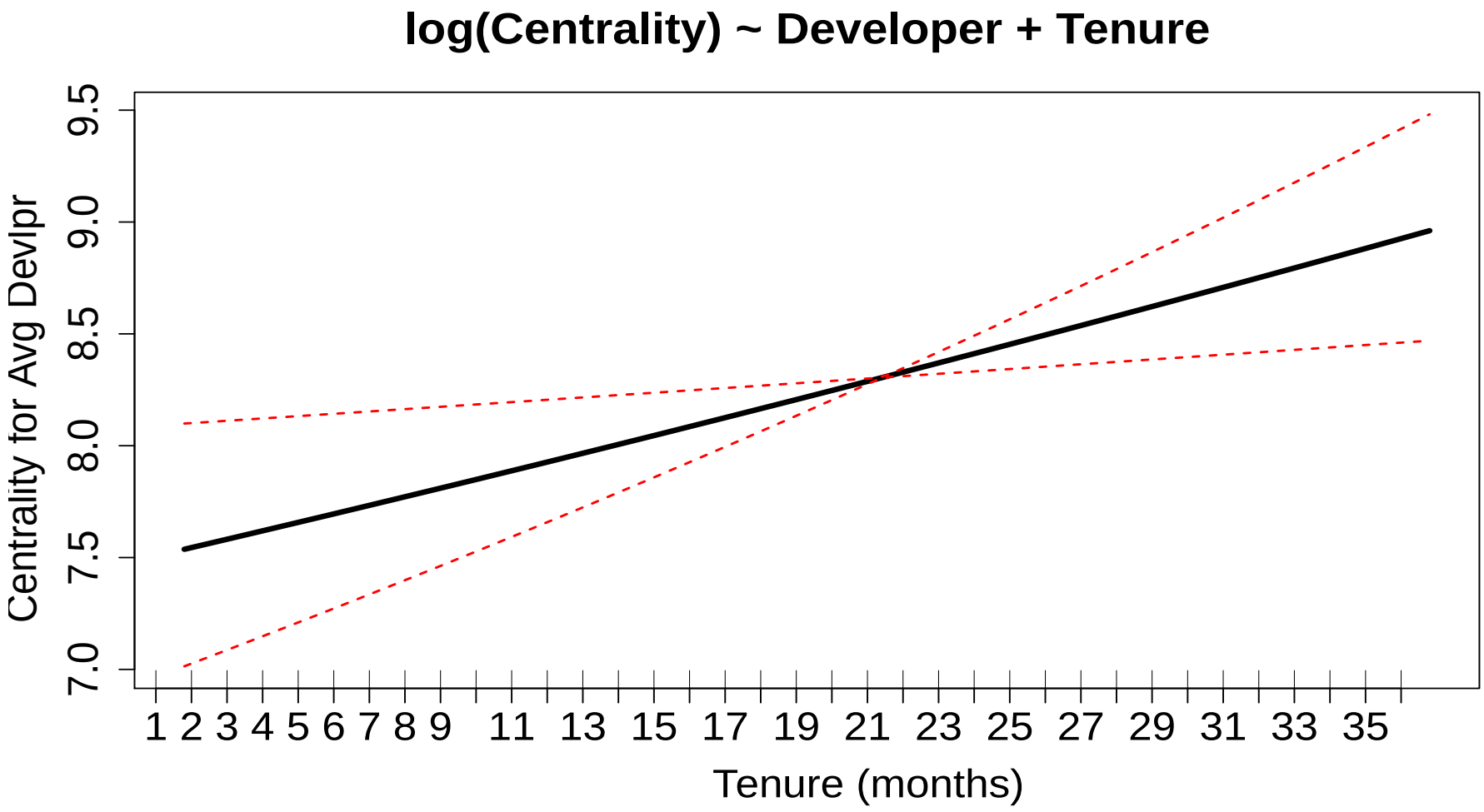
Modifications per month versus Tenure

7-10 months in smaller projects

Dimensions of project competency



But centrality of tasks continues to increase



Average task centrality (avg centrality of modules modified by the task) versus Tenure

Implications

- Conceptual results
  - Four dimensions of task difficulty
  - Four (corresponding) dimensions of task centrality
  - Measures for difficulty and centrality
  - Quantification of the growth of a developer's fluency
- Changes implemented in practice
  - Longer training periods in offshoring
  - Retaining some (or more of) experienced staff