Software Changes and Software Engineering

Audris Mockus and David Weiss

Software Technology Research Department

http://www.research.avayalabs.com/user/{audris,weiss}

Outline

- **Avaya Labs Research**

- **Goals and background**
  - Importance of software changes
- **Empirical understanding of software engineering**
  - Learning curves
  - Chunks
  - Who authors/maintains various pieces – Conway's homeomorphism
  - Risks of failures
- **Studying changes vs. studying code structure**

Empirical understanding of software engineering

- Learning curves
  - How long does it take a developer to become effective?
- Chunks
  - Is there a way to identify independent chunks of code?
- Who authors/maintains various pieces – Conway's homeomorphism
  - How does one identify “experts” in a particular area of code?
- Risks of failures
  - Given a change, what is the probability that it will fail in the field?
- Studying changes vs. studying code structure

Globalization problem

- Find a subset of software entities in site A that are the most appropriate for spare resources in site B
  - Will minimize future work dependencies between A and B
  - Will decrease existing work dependencies between A and B
  - Have appropriate amount of work

Current Practice

- Globalization decisions are made in an ad-hoc fashion
  - when resources become available
    - move the least important parts
    - move locality specific customization work
    - move releases in later maintenance stages
  - if something goes wrong - move it back to the main location
  - a lot of code bounces from location to location over time (lost productivity in learning new functionality)

Background

- Software is created incrementally, via changes recorded by a VCS

- A delta is addition and deletion of lines in a file

before:

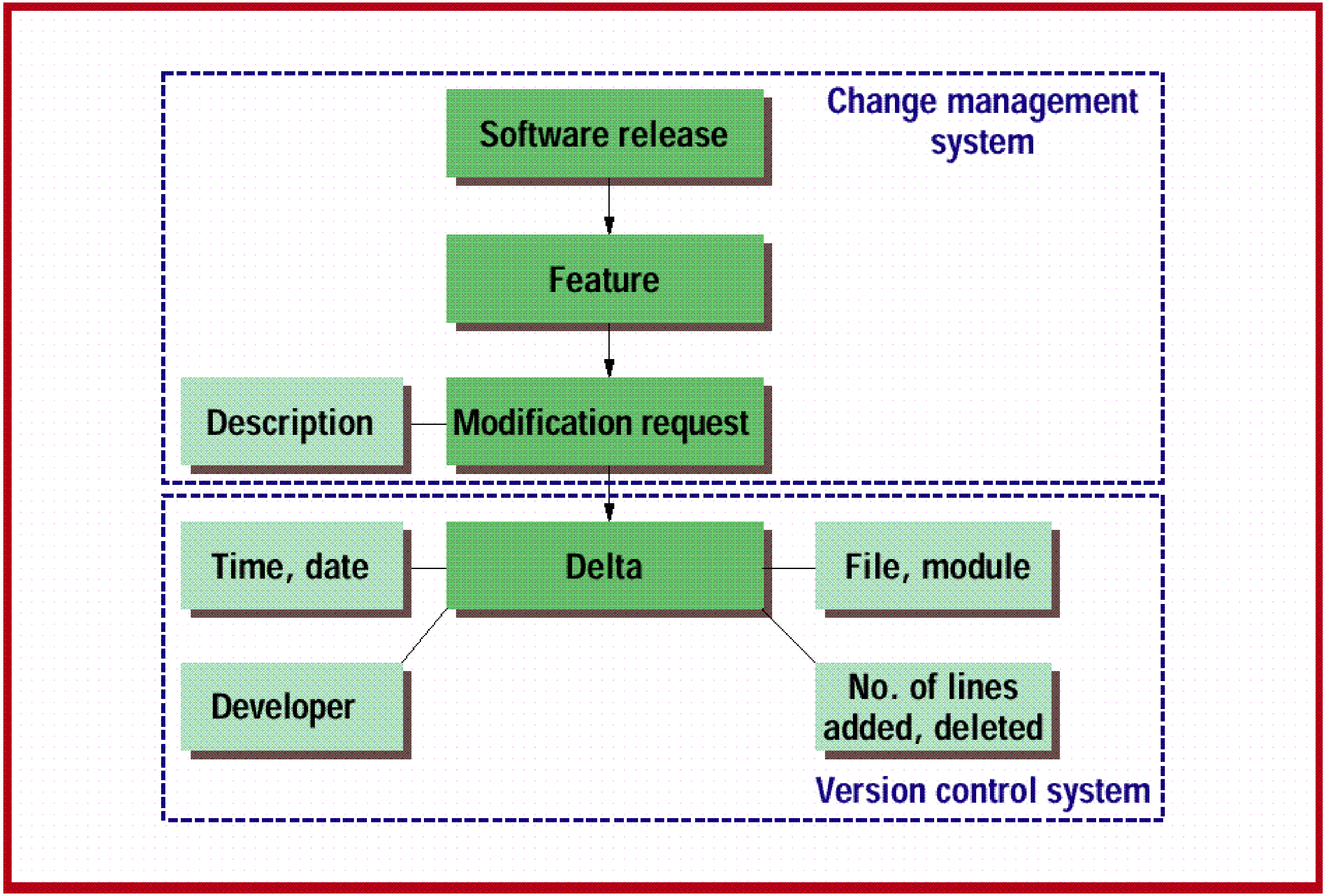
```

int i=N;
while (i)
     printf (“%d\n”,i--);
```

after:

```
// print N integers
int i=N;
while (N > 0 && i > 0)
     printf (“%d\n”,i--);
```

  - one line deleted
  - two lines added
  - two lines unchanged

Change Hierarchy



Basic Change Measures

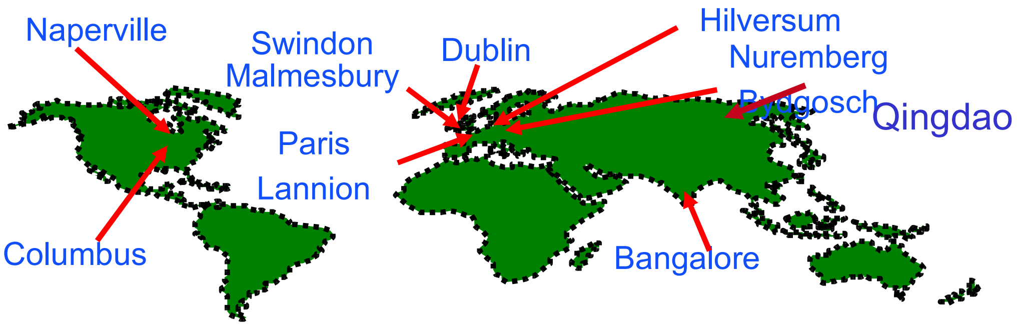
- Diffusion (# of subsystems, modules, files, developers)
- Size (# of lines added, deleted, and in the touched files)
- Diffusion & Size (# of deltas, MRs)
- Lead time (interval from start to completion)
- Purpose (Fix/New)
- Identity and experience (# of delta done in the past/recently/on a relevant part of the product) of creators

Advantages of Change Measures

+ obtainable for all projects using CM
+ nonintrusive– use existing data
+ fine grained – information at MR/delta level
+ complete – all parts of software are recorded
+ uniform – slowly change over time
+ massive – larger than surveys/project measures
+ unbiased – no observer effect
– data recorded for other purposes
– may need to use nontrivial datamining techniques

Some Projects Investigated

- Level 5 switching software product (140M lines added in 3M deltas over 16 years by 5K developers, in 5 primary locations on 4 continents)
- Call handling product (7M lines added in 200K deltas over 5 years by 110 developers, in 3 primary locations in 3 countries)
- OA&M Product (6M lines added in 100K deltas over 5 years by 350 developers, 3 primary locations in 3 countries)
- Wireless CH Product (14M lines in 140K deltas over 3 years by 340 developers, 5 primary locations in 5 countries)
- Optical network element product (1M lines added in 20K deltas over 2 years by 90 developers)
- Other
  - Apache (0.2M lines added in 15K deltas over 3 years by 15/300 developers)
  - Mozilla (6M lines added in 300K deltas over 3 years by 100/400 dev)



Audris Mockus and David Weiss

Distributing Work

- What work could be distributed?
- What are empirical dependencies in a system?
- Possible approaches:
  - Make work more independent
    - Fewer cross-site MRs ⇒fewer cross-person MRs ⇒less delay
  - Make developers more familiar with other people and their work
    - Speed up finding of relevant experts

Approach

To reduce the number multi-site work items (MRs) by re-assigning work among sites

1) Discretize code and work:

- Code units (CU) – subsystems or functional areas to be assigned
- Work units (WU) – MRs

2) Find subsets of CUs for each site based on criteria

- # of cross-site work units
- Effort to maintain assigned units

3) Evaluate a set of candidates

Finding Best Candidate

A simplified algorithm for reassigning code ownership between two sites

1. Choose initial set X of CUs randomly (constrained by effort)

2. Pick at random CU $y \in \neg X$ and do a) with probability τ or do b) with probability 1- τ

    a) **Add** y to X with probability 1 if adding decreases criteria, else add with probability $\mu<1$

    - reject if effort window is substantially violated

    b) **Exchange**: choose at random CU $z \in X$ and swap z and y with probability 1 swapping decreases criteria, else swap with probability $\pi<1$

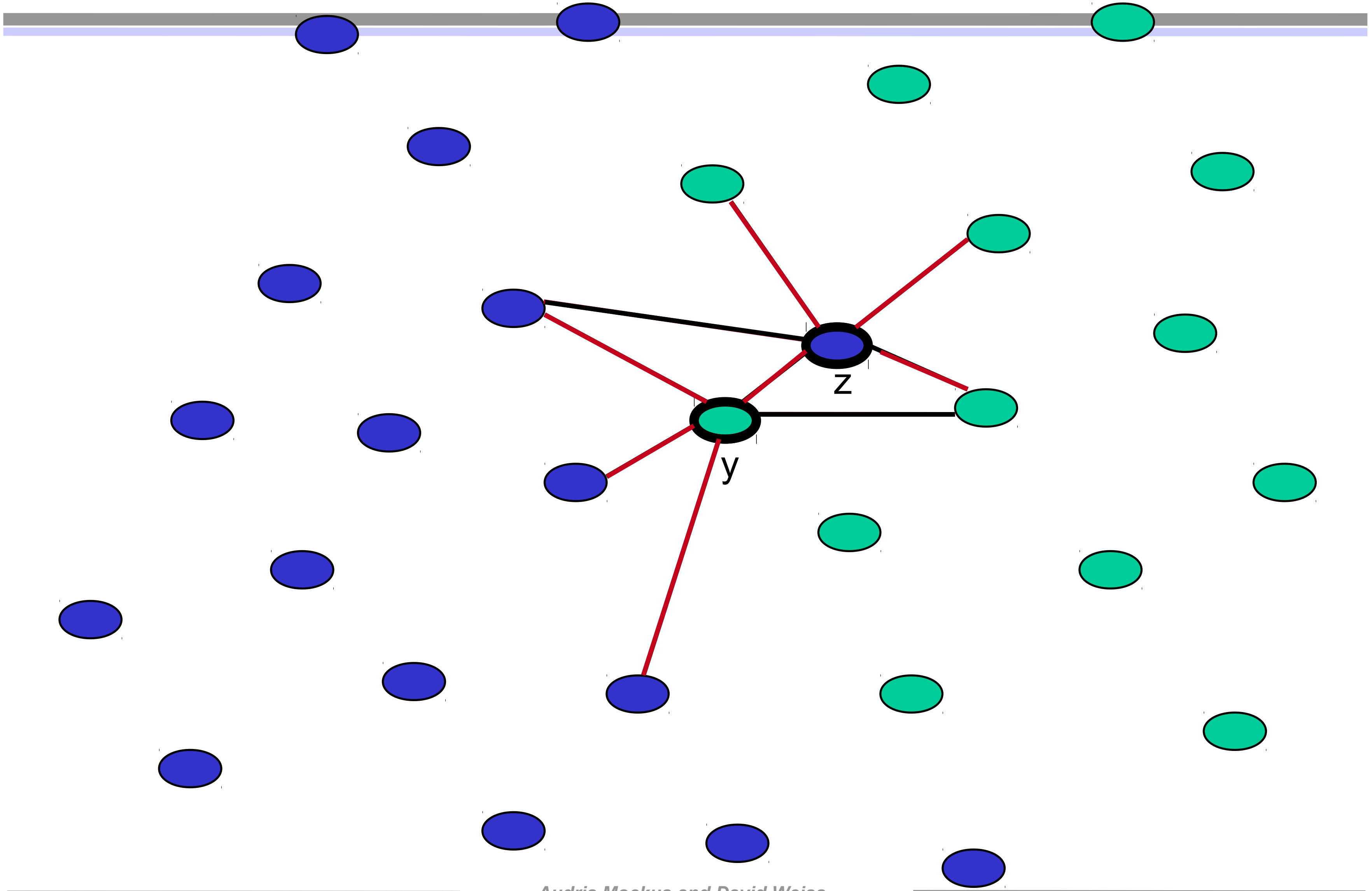
    - reject if effort window is substantially violated

3. record set X with best criteria for a number of effort ranges based on #MR touching X and $\neg X$

4. Go to step 2 or stop if number of iterations > N

$\Delta = 3$

Step A: Add node; $\Delta = 4$
Step B: Exchange nodes; $\Delta = 7$



Evaluation of Candidates

- Several candidate re-assignments of CUs

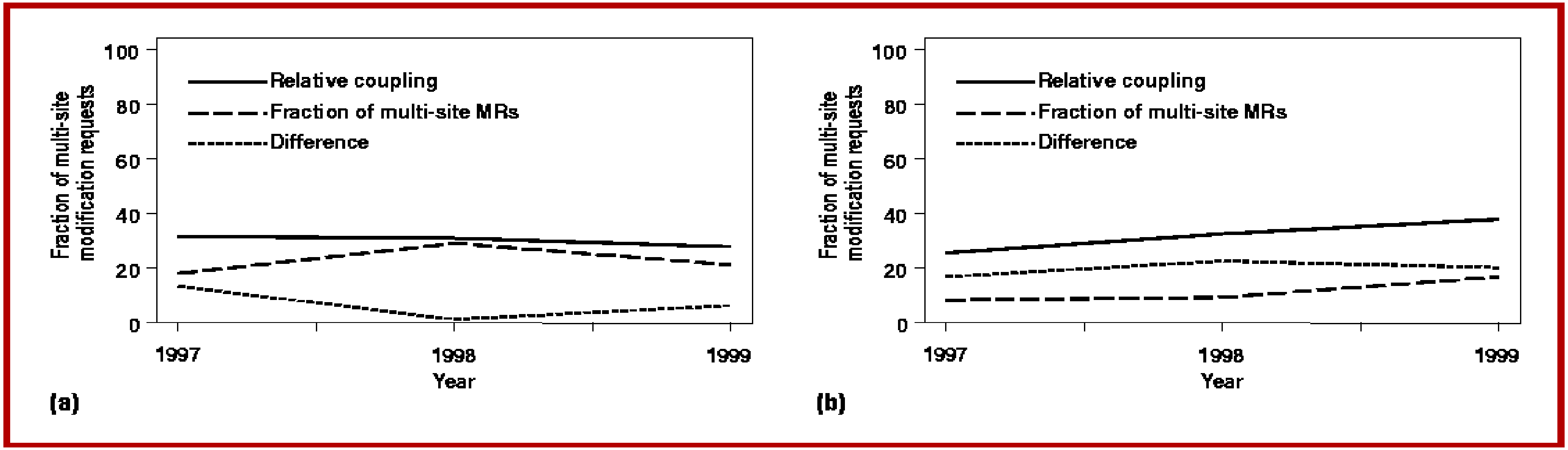
  a) Generated using algorithm

  b) Proposed by participants

- For each candidate present
  - Fraction of multi-site MRs
  - Effort trend (to predict effort needed in the future)
  - List of CUs
  - Interactive application providing instant feedback on alternative choices

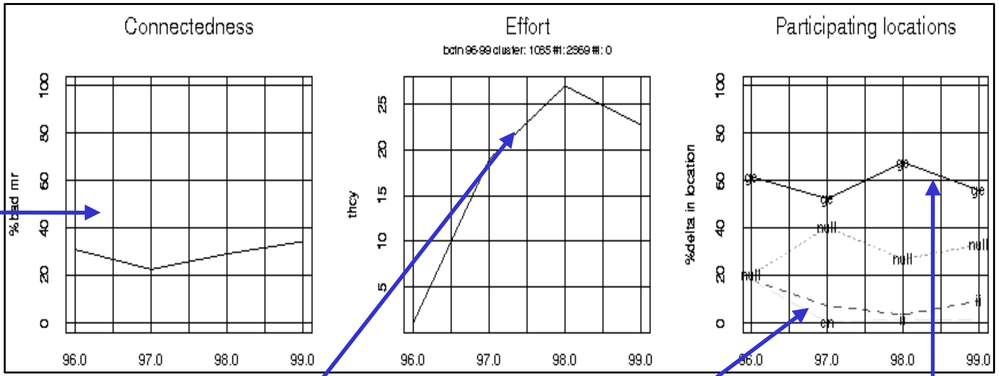
Evaluation Plots

Globalization candidates (a) and (b)



Details for candidate (b)

% of changes touching other code plotted over time



total effort

% of effort in England

in Germany

Interactive evaluation

Code Unit (CU) Hierarchy

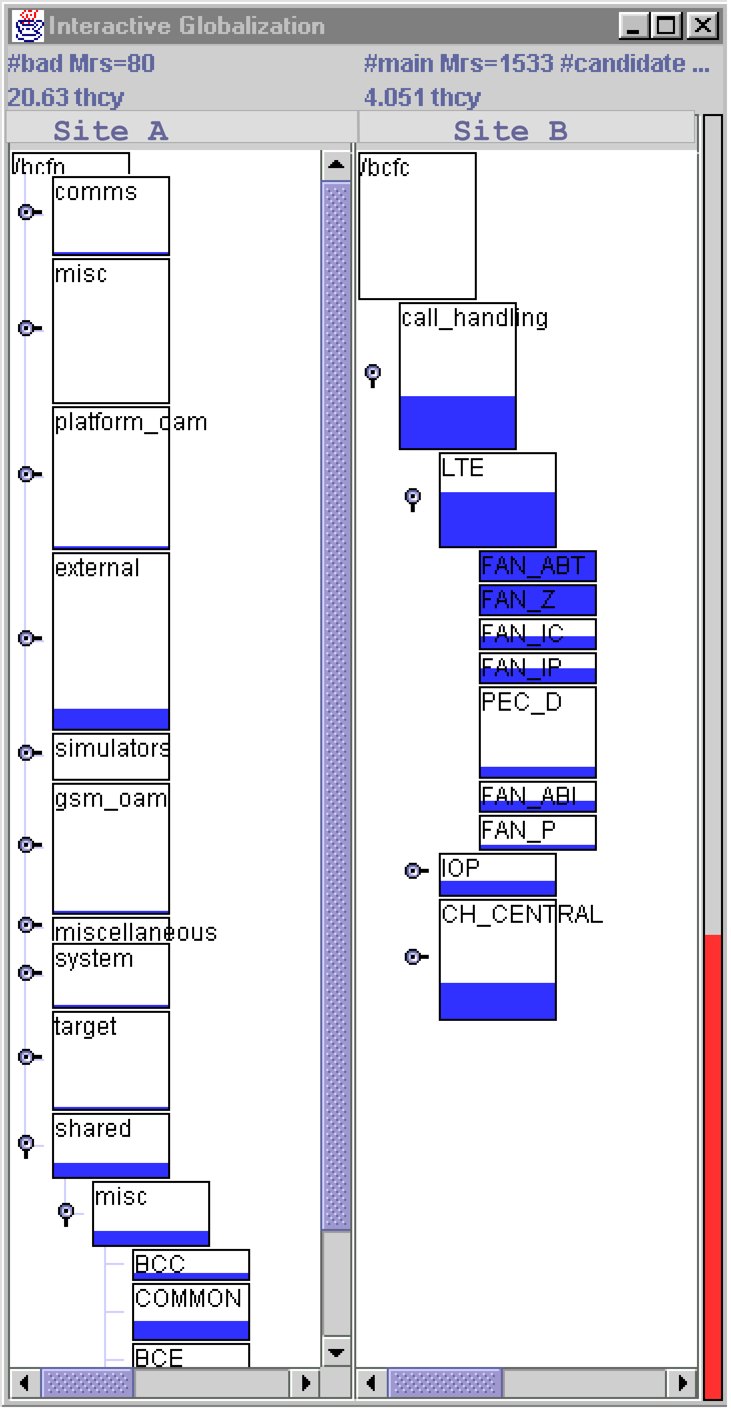
Site A (left) | Site B (right)

🟦 Fraction of MRs in CU crossing site boundary

🟥 Fraction of MRs crossing site boundary

Interactions:

Drag and drop desired CU from site A (left) to site B (right) or back



Lessons

- Other factors are important, e.g.,
  - the desirability of work, the criticality of work, the lack of desire to be dependent on transferred work, the loss of control.
- Other applications
  - assessing modularity problems
  - distributing work to contractors

For more details see: Audris Mockus and David M. Weiss. Globalization by chunking: a quantitative approach. IEEE Software, March 2001

Expertise Browser

- Goal and background
- How to reduce dependencies between sites?
  - Identifying independently changeable parts of code
- How to bring the sites “closer”
  - Finding expertise
  - Being aware of other’s work

Motivation

- **Conway's homeomorphism – how organization is mapped to code**
- **Common practical problems**
  - How to locate people/organizations who know that part of code?
  - How to make developers aware of changes that might impact their work?
- **Additional observations**
  - Only few people understand the entire SW system and they are typically in high demand
  - Each part of a system has several experts and each person is an expert on some parts of a system

Expertise (Experience) Measures

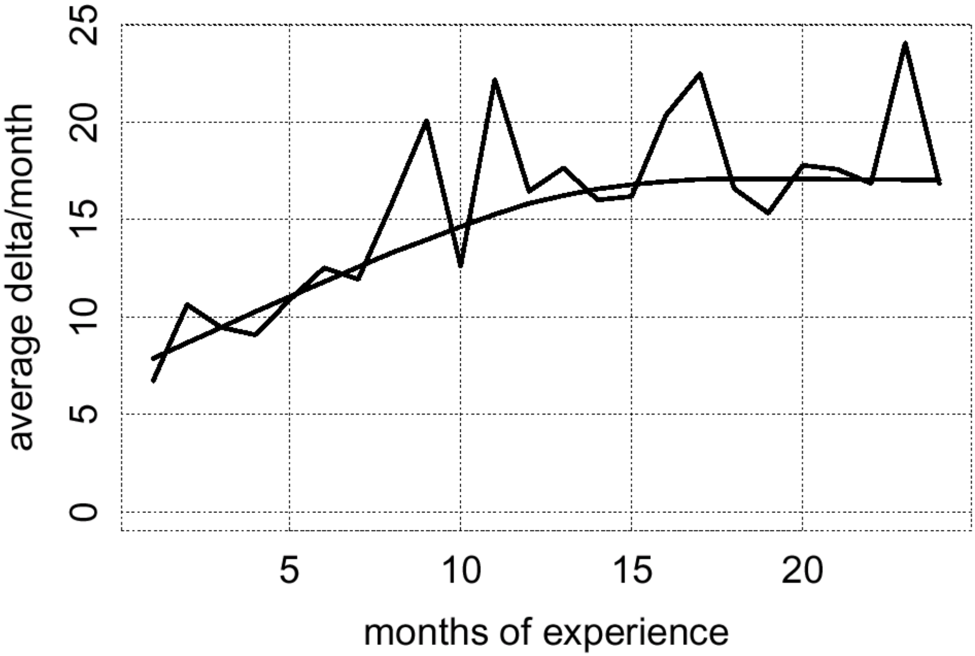
- Expertise: *Ability effectively to understand, enhance, fix, or test **a part** of a software system*

- Experience: *Amount of work (number of changes) performed on a **part** of a software system*

- Expertise ↑ Experience

- Expertise can be estimated

  directly from effort spent



Experience Atoms (EAs)

- Each change is a unit of experience or EA
  - Each EA identifies developer, date, file, change purpose (fix, new), problem report, language used, …
    - These properties are used to filter types of experience
- Example experience measures
  - Coding experience
    - effort spent on a CU
  - Testing experience
    - # of problem reports raised by a subject

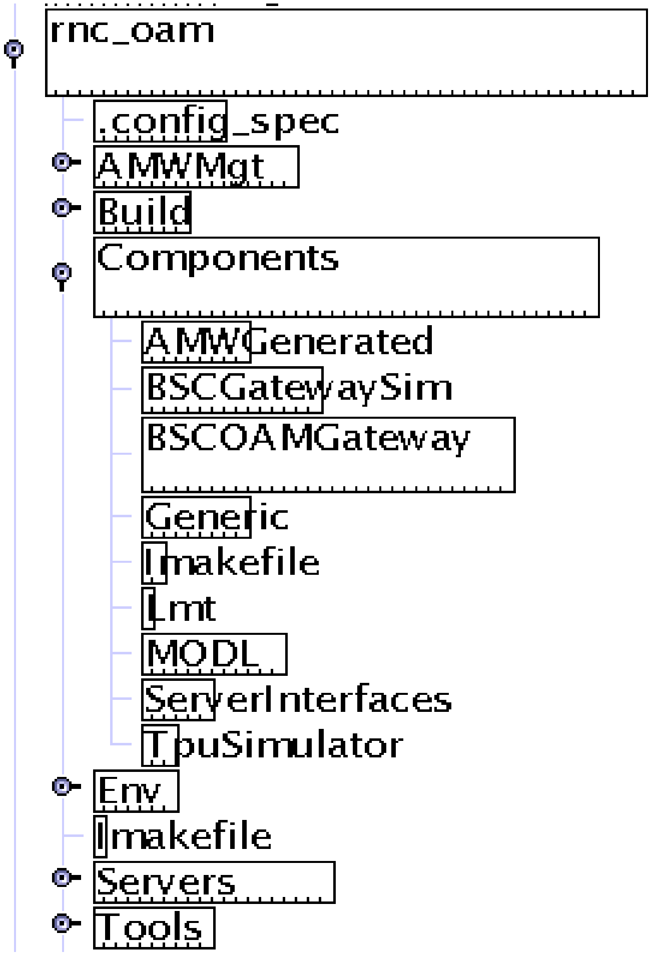
Audris Mockus and David Weiss

Expertise Browser

- Obtain and present relationships between code and people and organizations based on Experience Atoms (EAs) shared between CU and person
- User interface
  - Linked view paradigm (link by EAs)
  - Code, developer, organization, and detail views
  - Choosing CU shows people/orgs that are related
  - Selecting a person/org shows the fraction of work done on code modules and the persons contact info

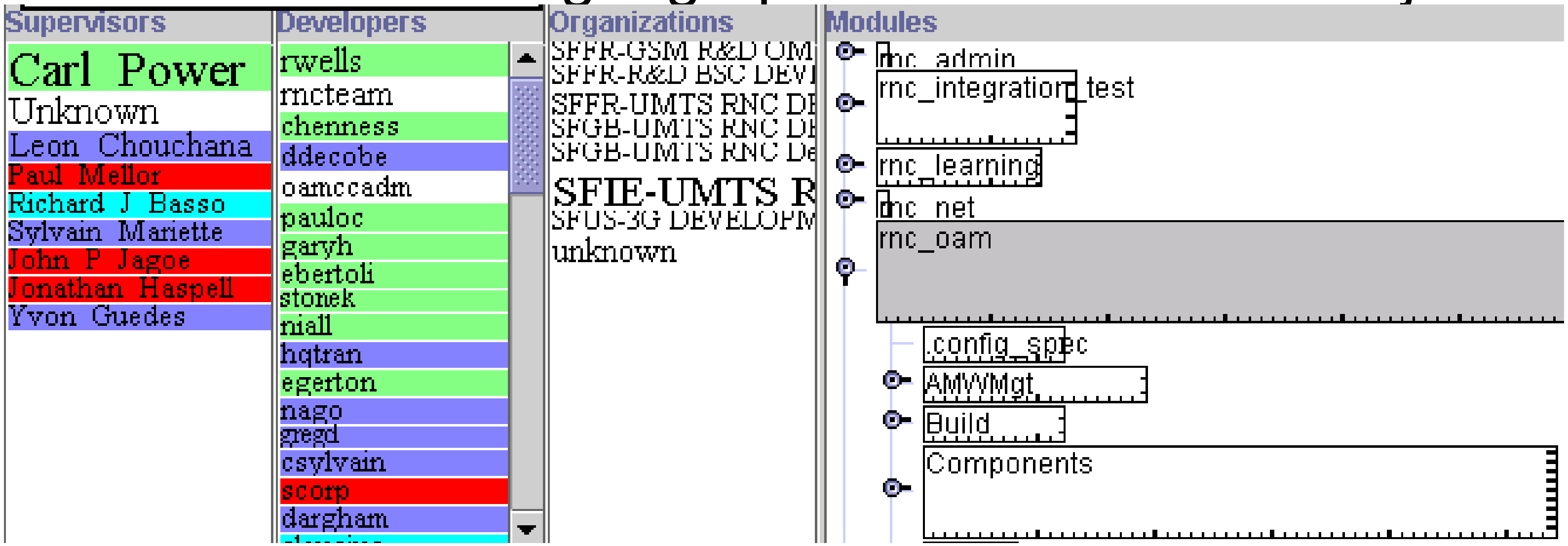
Code View

- An expandable tree normalized by changes (based on directories or subsystems/modules)
- Each node is a module/file or a set of modules/files
  - Height - sqrt(#of EAs/10)+font height
  - Width - 5 pixels per contributing subject



Expert Search

- Select a code unit to show experts
  - All developers, their supervisors, and organizations Ordered by expertise
    - Developers at the top are most relevant
    - Largest font reflects most experience
  - Color identifies geographic location of the subject



Supervisors

Carl Power
Unknown
Leon Chouchana
Paul Mellor
Richard J Basso
Sylvain Mariette
John P Jagoe
Jonathan Haspell
Yvon Guedes

Developers

rwells
rncteam
chenness
ddecobe
oamccadm
pauloc
garyh
ebertoli
stonek
niall
hqtran
egerton
nago
gregd
csylvain
scorp
dargham

Organizations

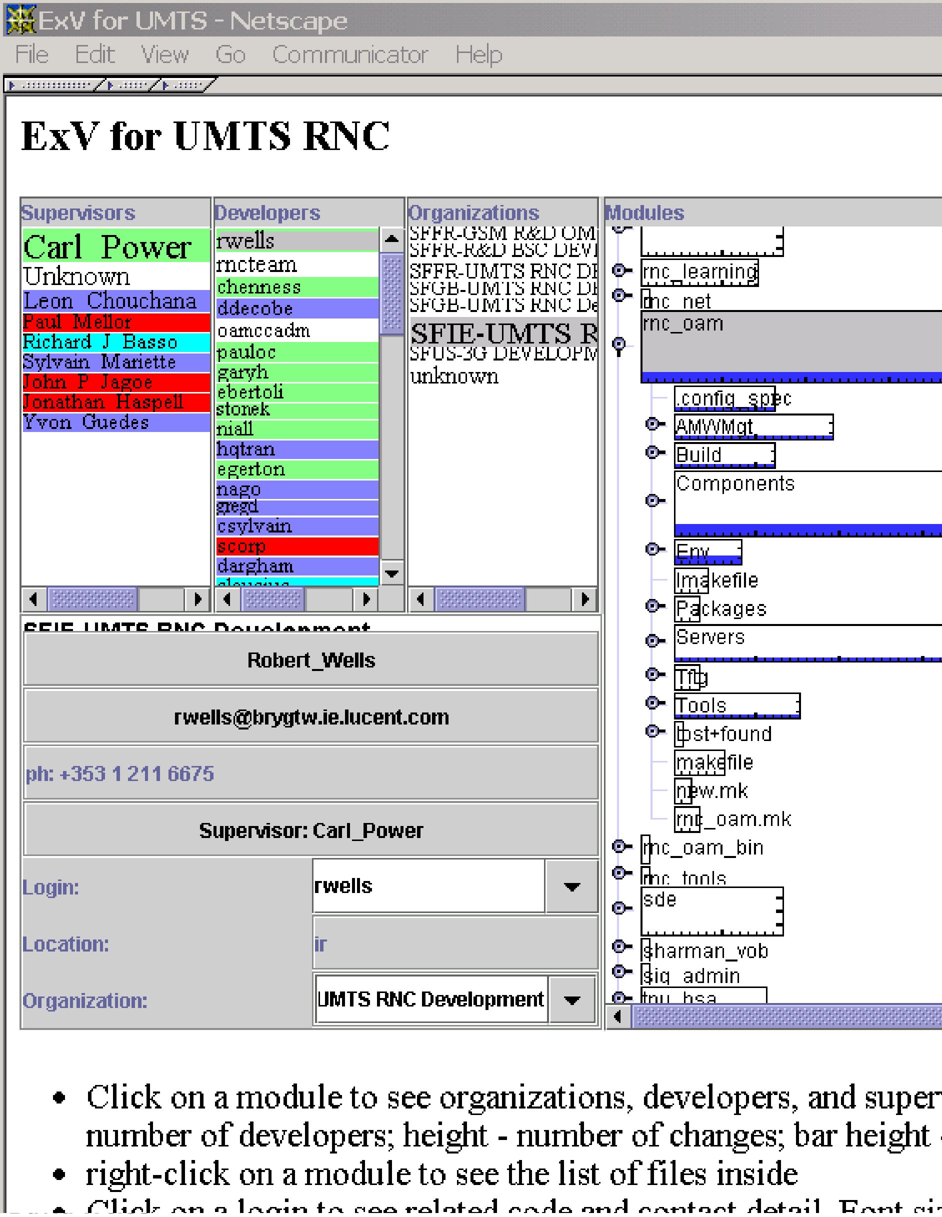
SFFR-GSM R&D OM
SFFR-R&D BSC DEV
SFFR-UMTS RNC DE
SFGB-UMTS RNC DE
SFGB-UMTS RNC De
SFIE-UMTS R
SFUS-3G DEVELOPM
unknown

Modules

rnc_admin
rnc_integration_test
rnc_learning
rnc_net
rnc_oam
.config_spec
AMWMgt
Build
Components

Resume View

- Select a person to show
  - Fraction of EAs for CUs
  - Contact info
- Select an org. to show
  - All developers in the organization/group
  - Fraction of EAs contributed by these developers for each CU



ExV for UMTS RNC

Supervisors	Developers	Organizations
Carl Power	rwells	SFFR-GSM R&D OM
Unknown	rncteam	SFFR-R&D BSC DEV
Leon Chouchana	chenness	SFFR-UMTS RNC DI
Paul Mellor	ddecobe	SFGB-UMTS RNC DI
Richard J Basso	oamccadm	SFGB-UMTS RNC De
Sylvain Mariette	pauloc	SFIE-UMTS R
John P Jagoe	garyh	SFUS-3G DEVELOPM
Jonathan Haspell	ebertoli	unknown
Yvon Guedes	stonek	
	niall	
	hqtran	
	egerton	
	nago	
	gregd	
	csylvain	
	scorp	
	dargham	

Modules

- rnc_learning
- rnc_net
- rnc_oam
  - .config_spec
  - AMWMgt
  - Build
  - Components
  - Env
  - Imakefile
  - Packages
  - Servers
  - Tftg
  - Tools
  - lost+found
  - makefile
  - new.mk
  - rnc_oam.mk
- rnc_oam_bin
- rnc_tools
- sde
- sharman_vob
- sig_admin
- tnu_hsa

SFIE-UMTS RNC Development

Robert_Wells

rwells@brygtw.ie.lucent.com

ph: +353 1 211 6675

Supervisor: Carl_Power

Login:	rwells
Location:	ir
Organization:	UMTS RNC Development

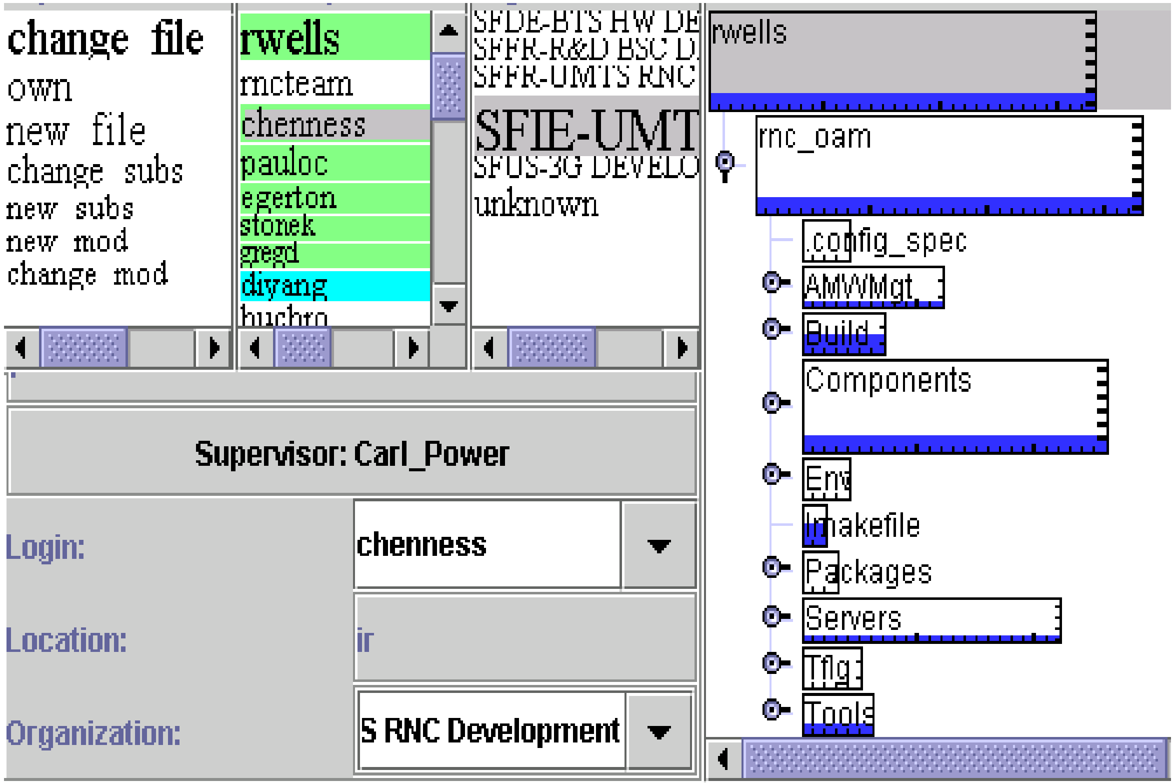
- Click on a module to see organizations, developers, and super
  number of developers; height - number of changes; bar height
- right-click on a module to see the list of files inside
- Click on a login to see related code and contact detail. Font si

Work Awareness

- Estimate persons “Home Area” using recent changes
- Define impact measures, e.g.,
  - Same line/file/module changed
  - Functions called are changed
- Determine/show others who do current work with potential impact

Who messed around my code?

- Individual view for rwells
  - Home Area: files and modules touched by rwells over last year
  - Changes by others done over last week on the same files and modules



change file
own
new file
change subs
new subs
new mod
change mod

rwells
rncteam
chenness
pauloc
egerton
stonek
gregd
diyang
buchro

SFDE-BTS HW DE
SFFR-R&D BSC D
SFFR-UMTS RNC
SFIE-UMT
SFUS-3G DEVELO
unknown

Supervisor: Carl_Power

Login: chenness

Location: ir

Organization: S RNC Development

Lessons

- ExB in three projects
  - 7M lines added in 200K deltas over 5 years by 110 developers
  - 6M lines added in 100K deltas over 5 years by 350 developers
  - 14M lines added in 140K deltas over 3 years by 340 developers
- Work awareness: just started to deploy
- Indications
  - New employees
  - New product (moved from other group)
- User feedback
  - New application to discover already raised problems for testers
  - Developers prefer directory view of the product/managers prefer subsystem/module view of the product
  - User interface improvements

Summary

- Business problems drive empiricism
- Focus on changes vs. focus on code structure
  - Data are widely available
    - Large development organizations need data to work effectively
    - Open source repositories
  - Analyst must understand limitations and potential of data
    - Initial time investment may be several years