Measuring Distributed Development

Audris Mockus

audris@avaya.com

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

Globally Distributed software development

Developers distributed over the world

Why?

Why SW development is and will be globally distributed?

National policies

Customer presence/local expertise

Easy/electronic transfer of the product

Available and inexpensive tele/data comm.

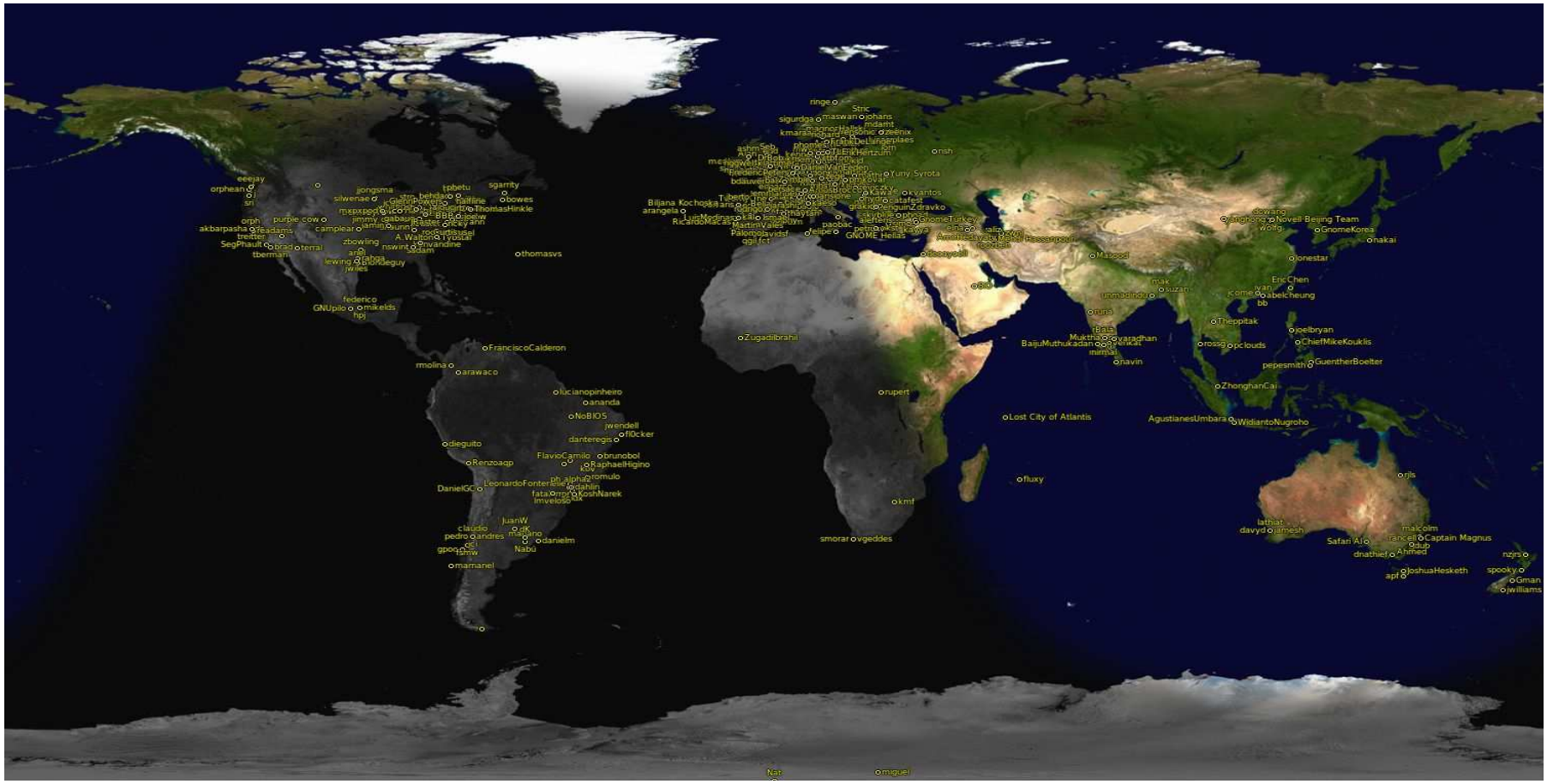
Lack of qualified IT workers in some countries/locations

Potential for lower costs

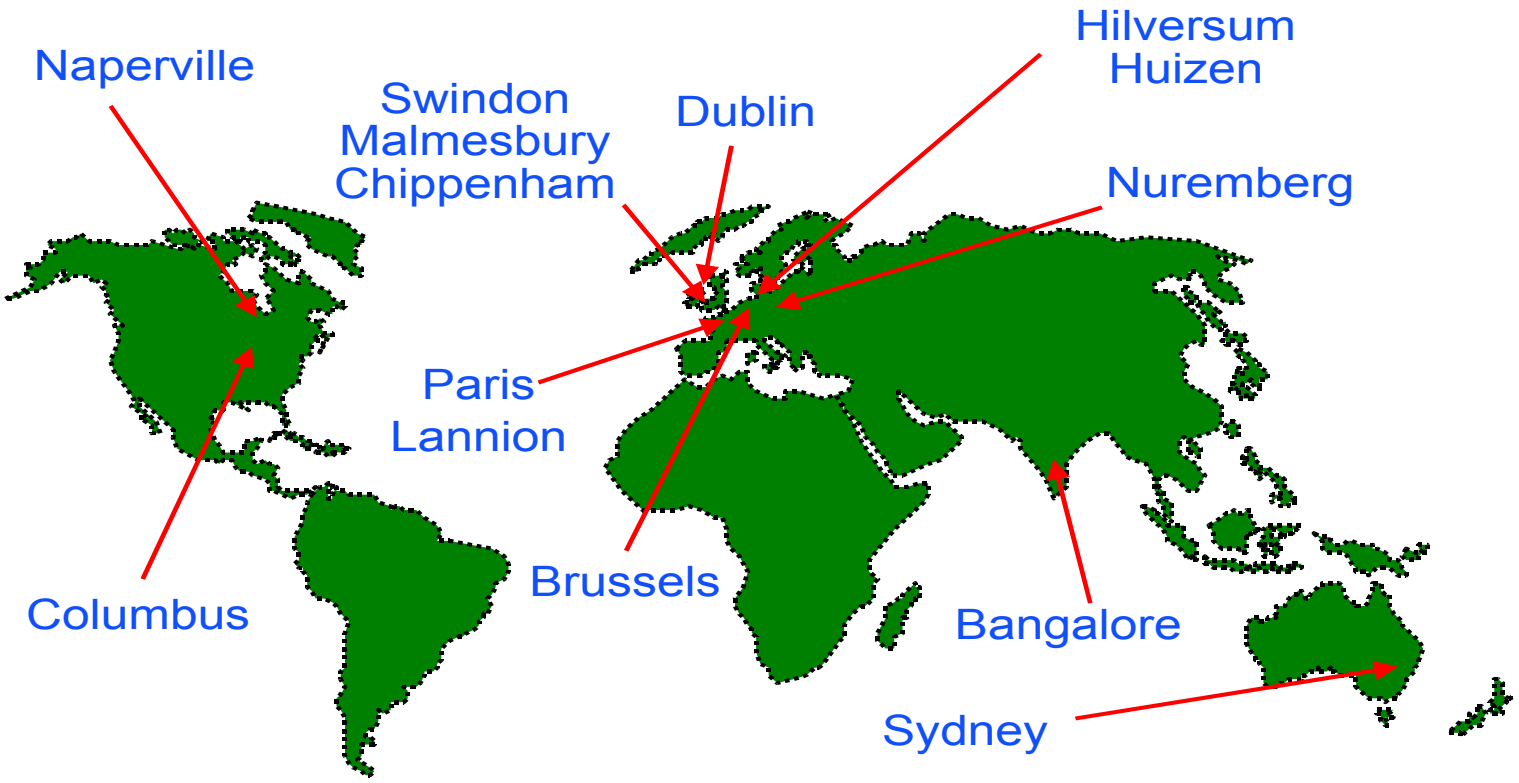
Potential for round-the-clock development

How many locations?

Gnome: several hundred developers



A large network switch: several thousand developers



7

How distributed developers communicate?

Use MR systems (e.g., Bugzilla, ClearQuest)

Use Version Control Systems

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

All development tasks are tracked via MRs

- Stages
  - Opened/Created
    - Developer decides to change code
    - User (or tester) experiences a fault with software and complains
    - New feature is started
  - Assigned: a person is assigned to solve the task
  - Submitted (code changed)/NoChanged/ReAssigned
  - Verified
- Example MR systems: Bugzilla

First Last Prev Next No search results available

Bug 616911 - Crash in load_books_thread at e-name-selector.c:115 Commit

Product:	Evolution
Component:	Contacts
Version:	2.31.x
Status:	UNCONFIRMED
Priority:	Urgent
Severity:	critical

Collapse All Comments - Expand All Comments

Akhil Laddha [reporter] 2010-04-27 04:51:11 UTC Description [reply] [-]

```
evolution 2.31.1

I double clicked on one of the mails in local draft folder and evolution
crashed.

(evolution:6921): GLib-GObject-WARNING **: invalid cast from `CamelIMAPXStore'
to `CamelOfflineStore'

(evolution:6921): GLib-GObject-WARNING **: invalid cast from `CamelIMAPXStore'
to `CamelOfflineStore'
[New Thread 0xa73ffb70 (LWP 6963)]
[New Thread 0xb2a11b70 (LWP 6964)]
[New Thread 0xaa951b70 (LWP 6965)]

(evolution:6921): GLib-CRITICAL **: g_array_append_vals: assertion `array'
failed

Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0xaa951b70 (LWP 6965)]
0xb7802a14 in load_books_thread (user_data=0x8a01cc8) at e-name-selector.c:115
115                     if (name_selector->priv->sections) {
(gdb) t a a bt
```

[+] Trace 221563

```
(gdb)
```

Akhil Laddha [reporter] 2010-06-01 06:23:09 UTC Comment 1 [reply] [-]

```
I get same crash *atleast* once per day :(
```


Milan Crha [developer] 2010-06-01 10:50:41 UTC Comment 2 [reply] [-]

Created an attachment (id=162450) [details]

```
debug eds patch

for evolution-data-server;

It'll print that the name_selector_gone on a console when it'll be freed during
the load_books_thread. If you see such print on evolution's console, do a
breakpoint and see when it gone:
  $ gdb evolution --ex "b name_selector_gone" --ex r
though I believe simple g_object_ref/unref on the name_selector in
load_books_thread will be the fix needed here. But let's see.
```

Milan Crha [developer] 2010-07-08 19:35:34 UTC Comment 3 [reply] [-]

```
You told me on IRC that you cannot reproduce the crash with this patch applied,
but you can still reproduce it without it. Could you try with a patch which
will have removed all the chunk
> @@ -112,11 +119,11 @@ load_books_thread (gpointer user_data)
because maybe it does the difference, please?
```

Additional Comments:

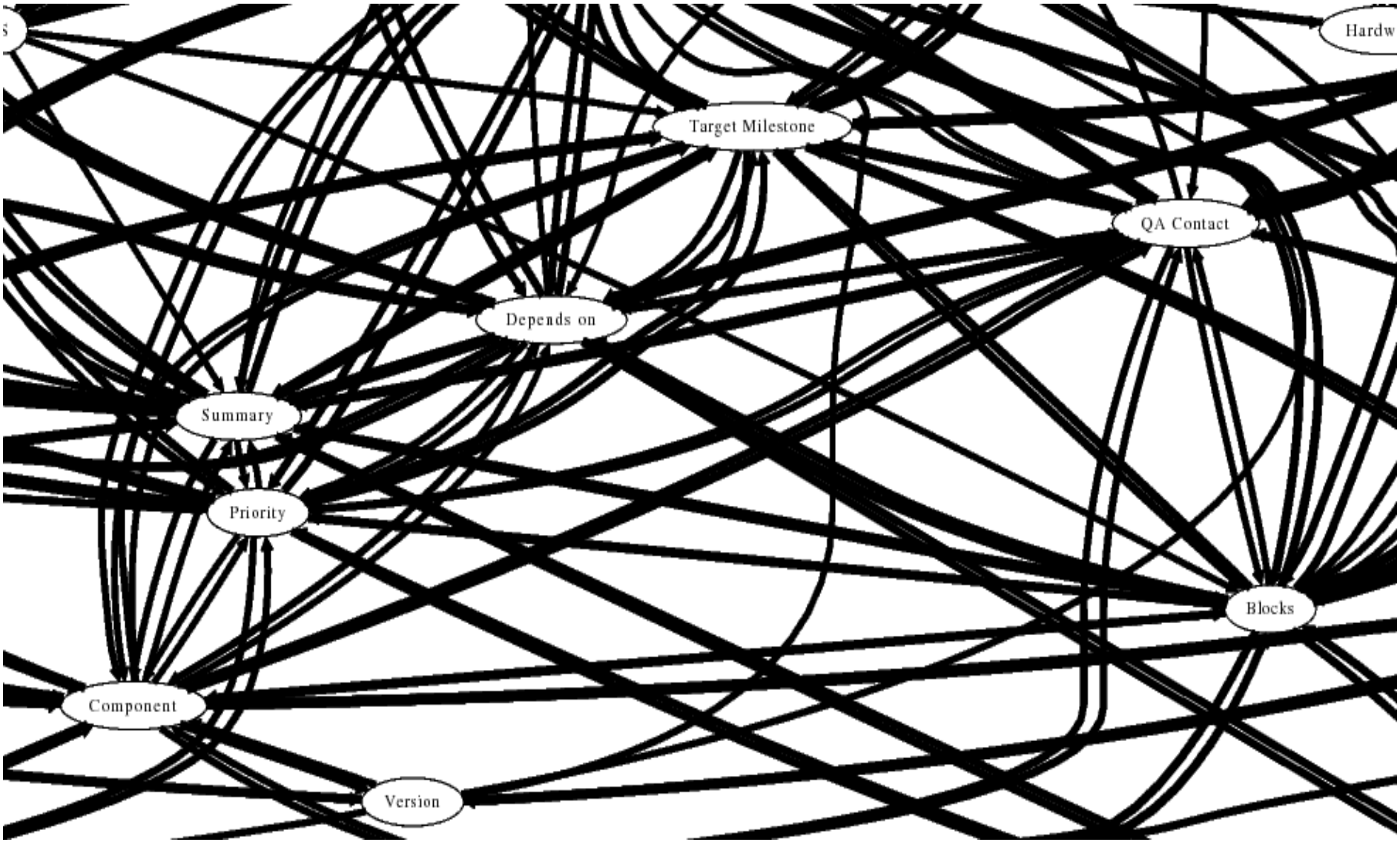
Status: UNCONFIRMED Commit

Attachments	
debug eds patch (1.94 KB, text/plain) 2010-06-01 10:50 UTC, Milan Crha	Details
Add an attachment (proposed patch, testcase, etc.)	View All

Status:	UNCONFIRMED
Product:	Evolution
Component:	Contacts
Version:	2.31.x
OS:	Linux
Importance:	Urgent critical
Target Milestone:	---
Assigned To:	evolution-addressbook-maintainers@ximian.com
QA Contact:	Evolution QA team
Whiteboard:	

Reported:	2010-04-27 04:51 UTC by Akhil Laddha
Modified:	2010-07-08 19:35 UTC (History)
CC List:	Add me to CC list 1 user (edit)
See Also:	
GNOME target:	---
GNOME version:	2.29/2.30

The transition frequency among MR states in Mozilla



Developers create software by *changes*

- All changes **are recorded**
- The product/code is simply a dynamic superposition of changes

Before:

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

After:

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

- one line deleted
- two lines added
- two lines unchanged
- Other attributes: date, developer, defect number, . . .
- Version Control System (VCS) track them, e.g., CVS/SVN/Git

cvsblame:

Who changed each line

Revision number

Bonsai version 1.3.9

CVS Blame

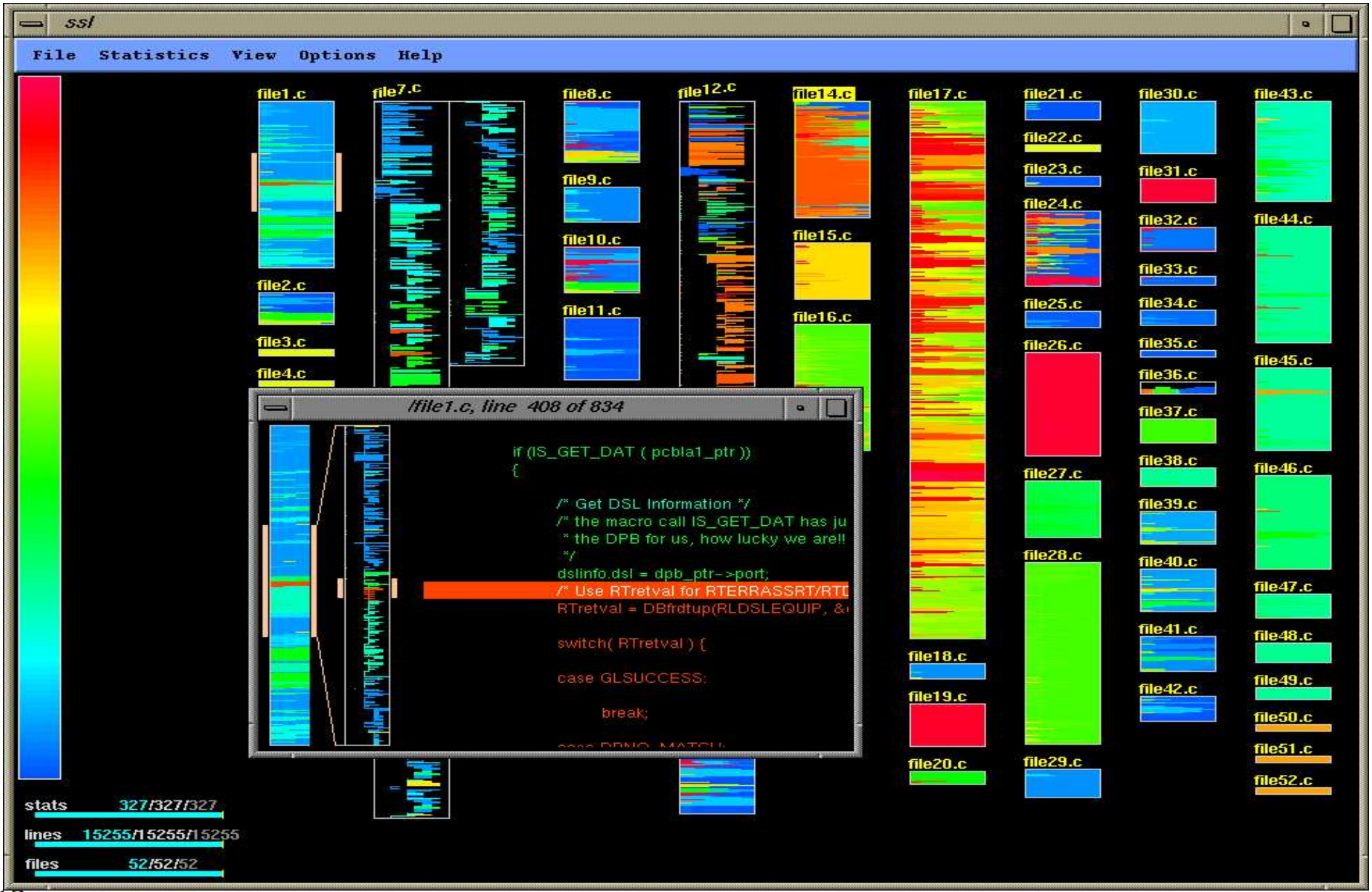
cm3/ m3-sys/ m3cc/ gcc/ gcc/ tree-scalar-evolution.h (1.2)

LXR: Cross Reference
Full Change Log

```
 1 hosking 1.1 /* Scalar evolution detector.
 2 jkrell  1.2    Copyright (C) 2003, 2004, 2005, 2007 Free Software Foundation, Inc.
 3 hosking 1.1    Contributed by Sebastian Pop <s.pop@laposte.net>
 4
 5             This file is part of GCC.
 6
 7             GCC is free software; you can redistribute it and/or modify it under
 8             the terms of the GNU General Public License as published by the Free
 9 jkrell  1.2 Software Foundation; either version 3, or (at your option) any later
10 hosking 1.1 version.
11
12             GCC is distributed in the hope that it will be useful, but WITHOUT ANY
13             WARRANTY; without even the implied warranty of MERCHANTABILITY or
14             FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
15             for more details.
16
17             You should have received a copy of the GNU General Public License
18 jkrell  1.2 along with GCC; see the file COPYING3.  If not see
19             <http://www.gnu.org/licenses/>.  */
20 hosking 1.1
21             #ifndef GCC_TREE_SCALAR_EVOLUTION_H
22             #define GCC_TREE_SCALAR_EVOLUTION_H
23
24 jkrell  1.2 extern tree number_of_latch_executions (struct loop *);
25             extern tree number_of_exit_cond_executions (struct loop *);
26             extern tree get_loop_exit_condition (const struct loop *);
27 hosking 1.1
28 jkrell  1.2 extern void scev_initialize (void);
29 hosking 1.1 extern void scev_reset (void);
30             extern void scev_finalize (void);
31             extern tree analyze_scalar_evolution (struct loop *, tree);
32             extern tree instantiate_parameters (struct loop *, tree);
33 jkrell  1.2 extern tree resolve_mixers (struct loop *, tree);
34 hosking 1.1 extern void gather_stats_on_scev_database (void);
35             extern void scev_analysis (void);
36 jkrell  1.2 unsigned int scev_const_prop (void);
37 hosking 1.1
38 jkrell  1.2 bool expression_expensive_p (tree);
39             extern bool simple_iv (struct loop *, tree, tree, affine_iv *, bool);
40
41             /* Returns the loop of the polynomial chrec CHREC.  */
42 hosking 1.1
43 jkrell  1.2 static inline struct loop *
44             get_chrec_loop (const_tree chrec)
45 hosking 1.1 {
46 jkrell  1.2   return get_loop (CHREC_VARIABLE (chrec));
47             }
48 hosking 1.1
49             #endif  /* GCC_TREE_SCALAR_EVOLUTION_H  */
```

Page configuration and help. Mail feedback to <tinderbox@elegosoft.com>.

SeeSoft: can show many files



Challenges and solutions?

Transfer of ownership

Chunking Code

Finding Expertise

Transfer of Ownership

Succession

- Definitions
  - *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*.
  - The receiving party: *follower*
  - The transferring party: *mentor*
  - In general, followers and mentors do not need to communicate with each other.
- Objective: measure *succession* and its impact.

How to measure succession?

- Projections
  - “Engaging” with the code often leads to changing the code
  - The chronological order of engagements by *mentors* and *followers* should be reflected in the temporal order of changes
- Therefore:
  - *Implicit team*: developers changing the same packages, files, methods, or lines
  - *Succession*: pairs of developers with the most clear succession signature
    - More shared code
    - Stronger chronological sequence

Data sources: most of Avaya's projects

- People: organizational Directory (LDAP) snapshots
  - Chronology: late 2001 and early 2003. Early 2004 until present: weekly extracts.
  - Attributes: personal ID, supervisor ID, department, location, phone, email
- People to login maps
  - Yellow pages (NIS), weekly extracts from three clusters
    - login to LDAP attributes, name
  - Proprietary problem reporting system (QQ), weekly extracts
    - login to name, email
- Version control systems
  - Chronology: 1990 until present, varies with project
  - Attributes: login, date, file

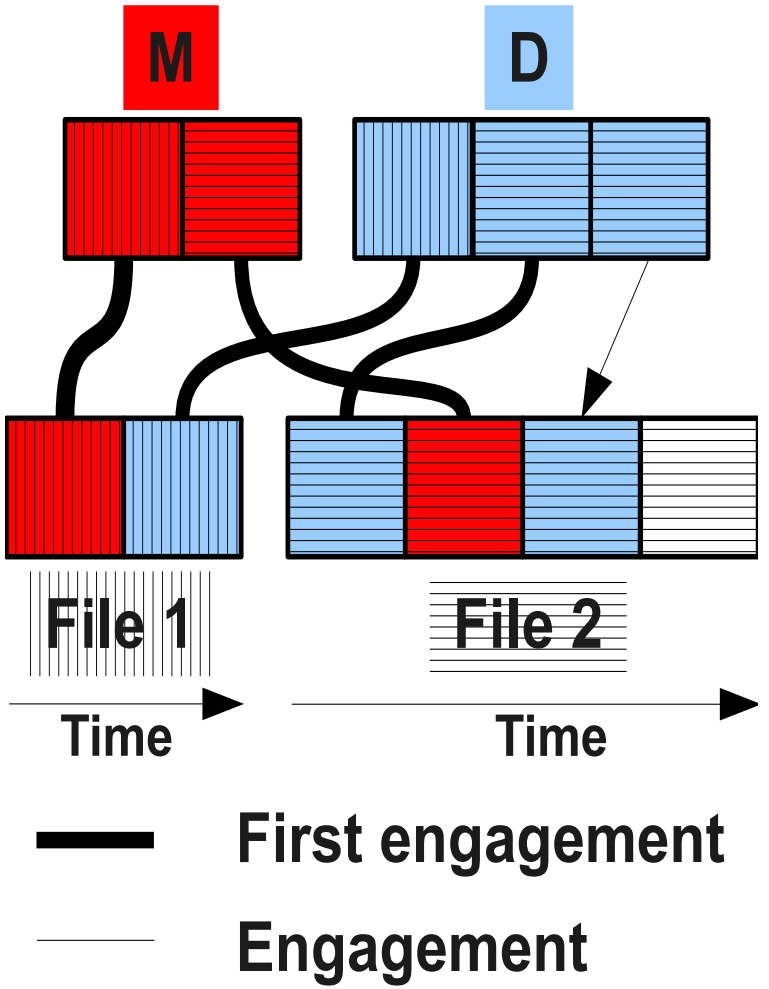
Illustration of succession signatures

Mentor m is:

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

$S(d, m)$: number of files touched by d and m weighted by fraction of file's changes made by d and m.

$$\begin{cases} S(d,m) & = & \frac{1+1}{2} \\ S(m,d) & = & \frac{2+1}{4} \end{cases} \implies m \; mentors \; d$$



Organizational socialization: succession

- Hypotheses
  - Offshoring succession is less informal/random $\Longrightarrow$ less innovation
  - Mentors with expertise dispersed over several products would provide mentorship that leads to more innovation
  - Mentors that transfer expertise of their secondary products would lead to less innovation by the followers
  - Mentors with more followers would have less innovative followers
  - Products with the oldest and largest code bases are likely to have lower productivity ratios
  - The effectiveness of expertise transfer increases over time as the organization improves its offshoring practices
- Custodial responses are likely to lead to a lower productivity ratio because followers will have to learn from mentor's example
- Productivity: number of delta (atomic changes) per month

The productivity ratio model

Table 1: $\log(Ratio) = Time + Offshore + Primary + Breadth + Size + \log(NFollow)$. 1012 mentor-follower pairs. Adj-$R^2 = 59$.

	Estimate	p-value	e^{est}	95%CI
Time of transfer	-0.01	0.31		
Offshoring	-0.63	0.00	$\frac{1}{2}$	$[0.42, 0.67]$
Primary expertise	-0.68	0.00	$\frac{1}{2}$	$[0.40, 0.64]$
Expertise breadth	-1.41	0.00	$\frac{1}{2}$	$[0.38, 0.64]$
Large prod.	-1.21	0.00	$\frac{1}{3}$	$[0.21, 0.42]$
Medium prod.	-0.46	0.00	$\frac{2}{3}$	$[0.52, 0.77]$
$\ln(NF)$	-0.53	0.00	$\sqrt{NF}$	

Practical implications

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

Chunking

Code

Marching orders

Find four candidate pieces of 10 to 20 technical headcount years each by April 4th.

Each piece must represent independent functionality.

P. L.

February 1999

History

In 1983 LNS development started in Naperville, IL, USE.

In 1985 LNS development started in Holland, and UK. Next major location was Poland in 1993.

Plans to start development in China, other locations

Current practice (1999)

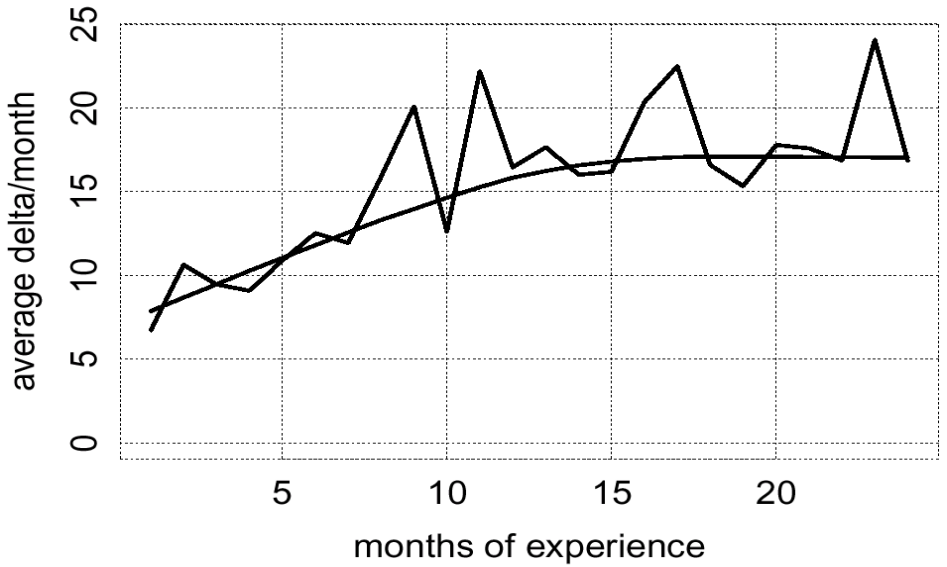
Globalization decisions are made in an ad-hoc fashion

- 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

Results of such decisions

Code bounces from location to location over time

(lost productivity in learning new functionality)



Impossible to learn: consequences of decisions are not known

Globalization problem: how to allocate development tasks?

Aid decision making process by finding a subset of functionality that are most appropriate for spare resources in location X:

- Is independently changeable (cohesive)
- Currently distributed across locations (high current cost)
- Matches expertise profile and spare capacity of location X

Define the problem

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

1) Discretize code and work:

   Code units (CU) — functional areas to be assigned

   Work units (WU) — MRs

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

   Number of cross-site work units

   Effort to maintain assigned units

3) Evaluate a set of candidates: Work Units and Code Units

Algorithm

Choose initial set X of CUs randomly so that it has 10 to 20 THCY

pick at random CU $y \in \neg X$ and do A) with probability

τ or do B) with probability $1 - \tau$

A) add y to X with probability 1 if adding decreases criteria, else add

with probability μ reject if THCY window is substantially violated

B) choose at random CU $z \in X$ and swap z and y with probability 1

swapping decreases criteria, else swap with probability π

reject if effort constraints are substantially violated

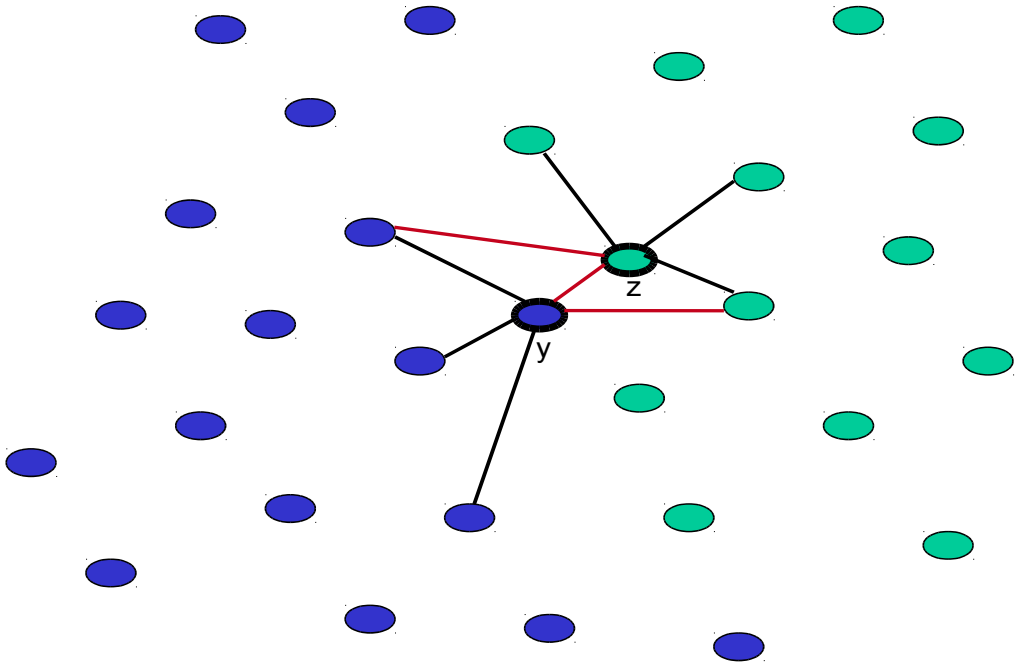
record set X with best criteria for a number of effort ranges

Criteria: # MR touching X and $\neg X$

X — green, $\neg X$ — blue,

If y is added to X, $\Delta = -2 + 3 = 1$.

If y is exchanged with z (y added to X, and z to $\neg X$): $\Delta = -1 - 1 + 3 + 3 = 4$



Evaluation of Candidates

Several candidate re-assignments of CUs

a) Generated using algorithm

b) Proposed by developers

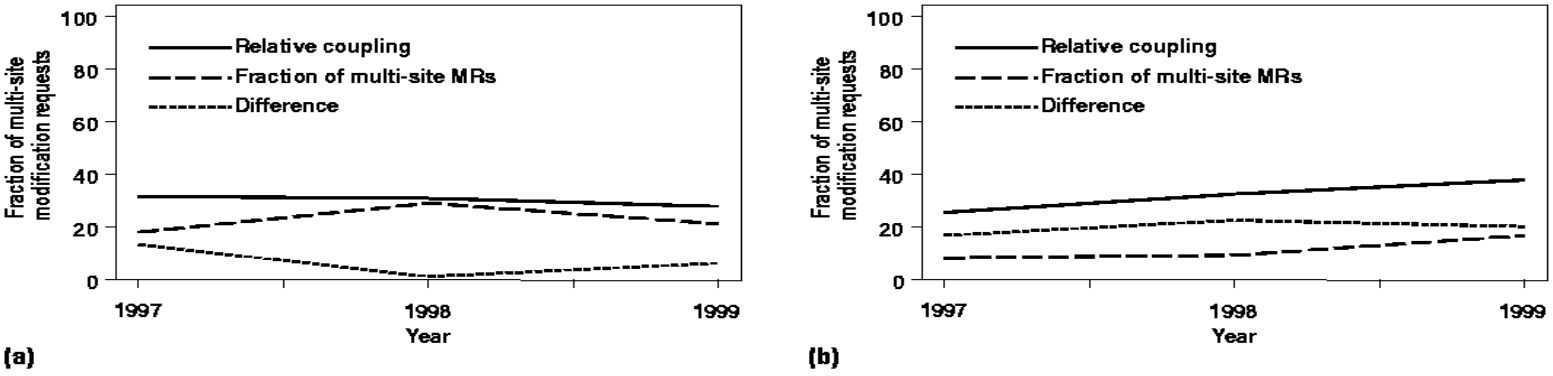
For each candidate present

Fraction of multi-site MRs

Effort trend (to predict effort needed in the future)

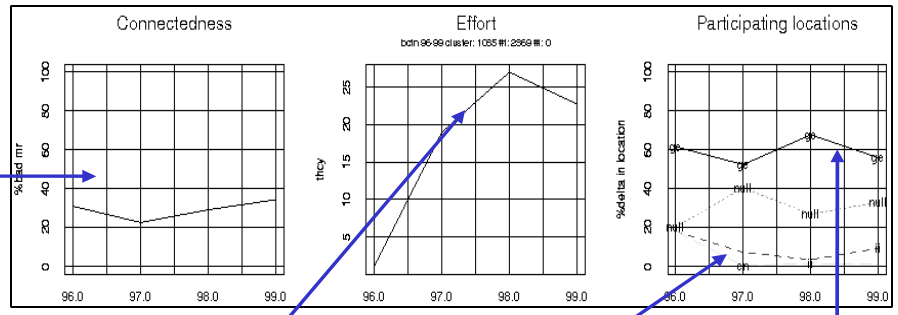
List of CUs

Globalization candidates (a) and (b)



Details for candidate (b)

% of changes touching other code plotted over time



Finding Expertise

How to locate people/organizations with specific experience?

Large software systems are complex

Few people understand entire system, but

Each part of a system has several experts

Each person is an expert on some parts of a system

Can one build a tool to perform tasks 1 and 2, i.e.,

How to find people who know a specific part of the code?

How to make developers aware of changes impacting their work?

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 to the code 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

Expertise Browser

Obtain and present relationships between code and people and organizations based on Experience Atoms (EAs) shared between CU and person

Expert Search

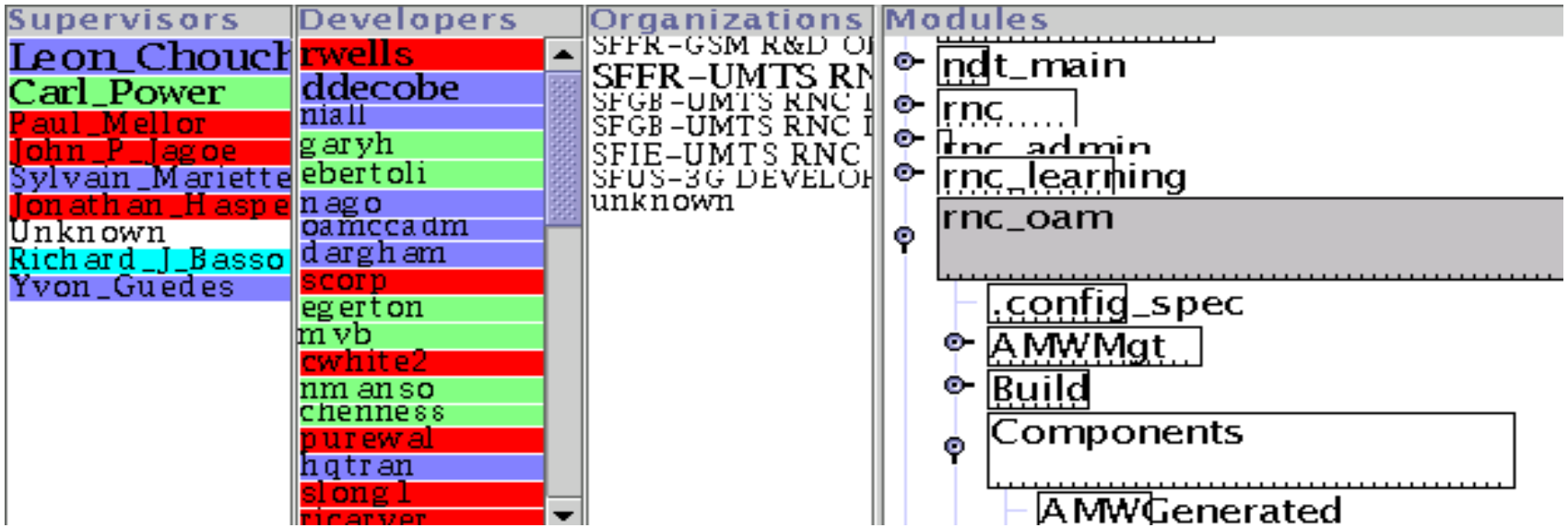
Select a code unit to show experts

All developers, their supervisors, and organizations Ordered by expertise

Developers at the top are most relevant

The largest font reflects most experience

Color identifies geographic location of the subject



Resume View

Select a person to show

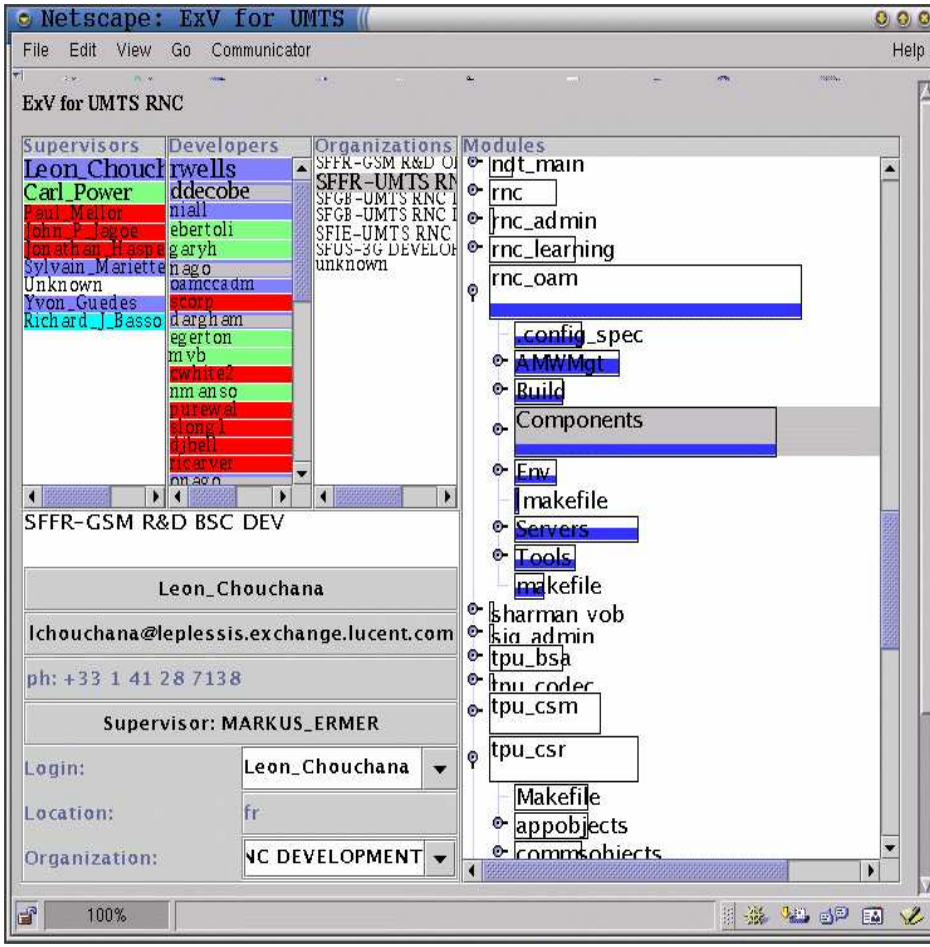
Fraction of EAs for CUs

Contact info

Select an org. to show

All developers in the organization/group

Fraction of EAs contributed by these developers for each CU



What have we done?

- Fundamental questions about human and collective nature
  - X is the study of *past human events and activities*
  - Y is the study of human **cultures** through the *recovery, documentation and analysis of* **material** *remains*
  - Z is the study of developer **cultures** and **behaviors** through the *recovery, documentation and analysis of* **digital** *remains*
- Is it X, Y, or Z?

Any wiser now?

Business problems need measurement?

Large and distributed development organizations need data to work effectively?

Bug tracking and VCS systems are a rich source of information about what people (developers) do?

and

perhaps they tell . . .

What **we** should do?

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 is interested in quantifying, modeling, and improving software development. He designs data mining methods to summarize and augment software change data, interactive visualization techniques to inspect, present, and control the development process, and statistical models and optimization techniques to understand the relationships among people, organizations, and characteristics of a software product. 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 in Avaya Labs Research. Previously he worked in the Software Production Research Department of Bell Labs.