Preprocessing CVS Data for Fine-Grained Analysis

Thomas Zimmermann\textsuperscript{1} and Peter Weißgerber\textsuperscript{2}

\textsuperscript{1} Saarland University
\textsuperscript{2} Catholic University of Eichstätt-Ingolstadt
Motivation

Tom Ball et al. “If your version control system could talk. . . ”

So, why is my CVS so silent?

1. CVS has limited query functionality and is slow.
   ⇒ Copy CVS into a database

2. CVS splits up changes on multiple files.
   ⇒ Infer transactions

3. CVS knows only files—but what about functions?
   ⇒ Detect fine-grained changes

4. CVS contains unreliable data which is noise.
   ⇒ Clean data

Preprocessing is the key to a talkative version control system.
Copy CVS into a Database

RCS file: /home/eclipse/org.eclipse.jdt.core/model/org/eclipse/jdt/core/IBuffer.java,v
Working file: ./../org.eclipse.jdt.core/model/org/eclipse/jdt/core/IBuffer.java
head: 1.17
branch:
locks: strict
access list:
symbolic names:
v_397: 1.16
v_396a: 1.16
... v_382: 1.15
JDK_1_5: 1.15.0.2
Root_JDK_1_5: 1.15
v_381: 1.15
...
keyword substitution: o
total revisions: 24; selected revisions: 24
description:
-----------------------------
revision 1.17
date: 2004/01/13 15:48:42; author: jnanneluc; state: Exp; lines: +1 -1
Updated copyrights to 2004
-----------------------------
revision 1.16
date: 2003/12/15 16:25:37; author: jnanneluc; state: Exp; lines: +15 -26
46040
-----------------------------
revision 1.15
date: 2003/05/26 16:13:24; author: pmulet; state: Exp; lines: +5 -1
branches: 1.15.2;
*** empty log message ***
-----------------------------
...
-----------------------------
revision 1.15.2.1
date: 2004/01/12 19:53:11; author: othomann; state: Exp; lines: +15 -26
Merge with HEAD

-----------------------------
Create incremental copies with cvs rdiff -s or cvs status.
**Infer Transactions: Time Windows**

All changes by the same developer, with the same message, made at the “same time” belong to one transaction.

**Fixed Time Window**

\[ \forall \delta_i : \forall \delta_j : \left| time(\delta_i) - time(\delta_j) \right| \leq T \]
Infer Transactions: Time Windows

All changes by the same developer, with the same message, made at the “same time” belong to one transaction.

**Fixed Time Window**
\[ \forall \delta_i : \forall \delta_j : |\text{time}(\delta_i) - \text{time}(\delta_j)| \leq T \]

\begin{align*}
\text{A:1.3} & \quad \text{B:1.2} & \quad \text{C:1.4} & \quad \text{D:1.3} & \quad \text{E:1.5} \\
\bullet & \quad \bullet & \quad \bullet & \quad \bullet & \quad \bullet
\end{align*}

same author + same message

**Sliding Time Window**
\[ \forall \delta_i : \exists \delta_j : |\text{time}(\delta_i) - \text{time}(\delta_j)| \leq T \]

\begin{align*}
\text{A:1.3} & \quad \text{B:1.2} & \quad \text{C:1.4} & \quad \text{D:1.3} & \quad \text{E:1.5} \\
\bullet & \quad \bullet & \quad \bullet & \quad \bullet & \quad \bullet
\end{align*}

same author + same message
Infer Transactions: Time Windows

All changes by the same developer, with the same message, made at the “same time” belong to one transaction.

Fixed Time Window

\[ \forall \delta_i : \forall \delta_j : |\text{time}(\delta_i) - \text{time}(\delta_j)| \leq T \]

Sliding Time Window

\[ \forall \delta_i : \exists \delta_j : |\text{time}(\delta_i) - \text{time}(\delta_j)| \leq T \]
**Infer Transactions: Time Windows**

All changes by the same developer, with the same message, made at the “same time” belong to one transaction.

**Fixed Time Window**

\[ \forall \delta_i : \forall \delta_j : |time(\delta_i) - time(\delta_j)| \leq T \]

**Sliding Time Window**

\[ \forall \delta_i : \exists \delta_j : |time(\delta_i) - time(\delta_j)| \leq T \]
**Infer Transactions: Time Windows**

All changes by the same developer, with the same message, made at the “same time” belong to one transaction.

**Fixed Time Window**

\[ \forall \delta_i : \forall \delta_j : |time(\delta_i) - time(\delta_j)| \leq T \]

**Sliding Time Window**

\[ \forall \delta_i : \exists \delta_j : |time(\delta_i) - time(\delta_j)| \leq T \]
Infer Transactions: Time Windows

All changes by the same developer, with the same message, made at the “same time” belong to one transaction.

Fixed Time Window

\[ \forall \delta_i : \forall \delta_j : |time(\delta_i) - time(\delta_j)| \leq T \]

Sliding Time Window

\[ \forall \delta_i : \exists \delta_j : |time(\delta_i) - time(\delta_j)| \leq T \]

All changed files within one transaction have to be different.
Infer Transactions: Commit Mails

All changes listed in a commit mail belong to one transaction.

CVSROOT: /cvs/gcc
Module name: gcc
Changes by: zack@gcc.gnu.org 2004-05-01 19:12:47

Modified files:
gcc/cp : ChangeLog decl.c

Log message:
* decl.c (reshape_init): Do not apply TYPE_DOMAIN to a VECTOR_TYPE. Instead, dig into the representation type to find the array bound.

Patches:
http://.../cvsweb.cgi/gcc/gcc/cp/ChangeLog.diff?...&r2=1.4042
http://.../cvsweb.cgi/gcc/gcc/cp/decl.c.diff?...&r2=1.1204

Commit mails for GCC: http://gcc.gnu.org/ml/gcc-cvs/

Not every project provides useful commit mails.
**Infer Transactions: Evaluation**

We inferred transactions for 3 years GCC using commit mails.

**Maximal Duration of a Commit**

21:17 minutes for “merged with ra-merge-initial” (5,910 files)

⇒ Sliding time windows are superior to fixed ones.
**Infer Transactions: Evaluation**

We inferred transactions for 3 years GCC using commit mails.

**Maximal Duration of a Commit**

21:17 minutes for “merged with ra-merge-initial” (5,910 files)

⇒ Sliding time windows are superior to fixed ones.

**Maximal Distance between two subsequent Checkins**

Depends on file size, RCS file size, and # of revisions. For almost all files below 3:00 minutes. Two exceptions: gcc/libstdc++-v3/configure, gcc/gcc/ChangeLog

⇒ Time windows should be at least 3:00 minutes.
Infer Transactions: Evaluation

We inferred transactions for 3 years GCC using commit mails.

Maximal Duration of a Commit
21:17 minutes for “merged with ra-merge-initial” (5,910 files)
⇒ Sliding time windows are superior to fixed ones.

Maximal Distance between two subsequent Checkins
Depends on file size, RCS file size, and # of revisions.
For almost all files below 3:00 minutes. Two exceptions: gcc/libstdc++-v3/configure, gcc/gcc/ChangeLog
⇒ Time windows should be at least 3:00 minutes.

Minimal Distance between two similar Commits
Bad news: 0:02 minutes for “Mark ChangeLog”
Good news: All similar commits were really related.
⇒ Time windows have no upper bound (no duplicate files!)
Detect Fine-Grained Changes

What building blocks (e.g., functions, classes, sections, etc.) have been changed between two revisions?

Rev. $r_1$

```c
void A()
{
  ...
}
void B()
{
  ...
}
void C()
{
  ...
}
void D()
{
  ...
}
void E()
{
  ...
}
```

Rev. $r_2$

```c
void A()
{
  ...
}
void F()
{
  ...
}
void B()
{
  ...
}
void D()
{
  ...
}
void E()
{
  ...
}
```
Detect Fine-Grained Changes

What building blocks (e.g., functions, classes, sections, etc.) have been changed between two revisions?

1. Parse $r_1$ for entities

```
void A(){
    ...
}
void B(){
    ...
}
void C(){
    ...
}
void D(){
    ...
}
void E(){
    ...
}
```

```
B() i=42;
C()
D()
E()
```

2. Parse $r_2$ for entities

```
void A(){
    ...
}
void F(){
    ...
}
void B(){
    ...
}
void D(){
    ...
}
void E(){
    ...
}
```

```
A()
F()
B() i=23;
D()
E()
```
Detect Fine-Grained Changes

What building blocks (e.g., functions, classes, sections, etc.) have been changed between two revisions?

1. Parse $r_1$ for entities

2. Parse $r_2$ for entities

3. Compare matching entities
**Noise: Large Transactions**

Large transactions are usually outliers:

- “Change #include filenames from `<foo.h>` [sigh] to `<openssl.h>`.” (552 files, OPENSSL)

- “Change functions to ANSI C.” (491 files, OPENSSL)

**Solution:** Ignore all transactions with size above N.
Noise: Merge Transactions

Changed files:
- A, B
- C, D
- E, F
- G, H

Branch Point

Merge Point

Branch can continue

More merges for a single branch are possible

○ = Commit/Transaction
Merges are *noise* for two reasons:

1. Merges contain unrelated changes — e.g. $B$ and $C$
2. Merges duplicate related changes — e.g. $A$ and $B$
Noise: Merge Transactions

Two Solutions:

- The Fischer/Pinzger/Gall heuristic (ICSM 2003).
- Suspect & Verify approach based on log messages.

Problem:
“New isMerge(), isMergeWithConflicts(), and . . . ”
Lessons Learned

★ Databases simplify the exploration of CVS.
★ Sliding time windows are superior to fixed ones.
★ Length of time windows should be within 3 and 5 minutes.
★ Fine-grained analyses are feasible and worth while.
★ Take a look at the ECLIPSE framework for comparing files: org.eclipse.compare.structuremergeviewer
★ Merges are dirty transactions and difficult to recognize.

Preprocessing is the key to any good and reliable analysis.