Metrics of Change

Abram Hindle

Software Engineering Group

Department of Computer Science

University of Victoria

abez@uvic.ca

February 4, 2005

This Presentation

- What am I going to cover?
  - Source Code Repositories
  - softChange
  - CheeseGod
  - The Data
  - Non-Derived Metrics
  - Derived Metrics
  - Summary

Software Evolution

- Why study software in this manner?
  - Programmers are not always available for interview.
  - Provide historical evidence about software.
  - Correlate Project History to the Source Code.
  - Verify assertions about the project’s development.

Source Code Repositories

- Products
  - OSS: CVS, Subversion, Arch, Darcs
  - Proprietary: Clear Case, Perforce, Source Safe, BitKeeper
- Functionality
  - Revisions
  - Branches
  - Concurrency
  - Configuration Management

CVS

- Why CVS?
  - Defacto Standard for Open Source projects.
  - Many mature Open Source Projects have open repositories to study.
  - Learn about Open Source Software Development processes.
- CVS Operations
  - commit
  - update
  - checkout
- We attempt to track CVS Commits by grouping revisions.

softChange

- What is softChange ?
  - softChange is a collection of applications that work together in order to further study the software evolution of a project.
  - softChange elaborates on data provided from many sources to enable an accurate description of the evolution of a project.
  - softChange helps answer common questions maintainers, developers and administrators have about a project.

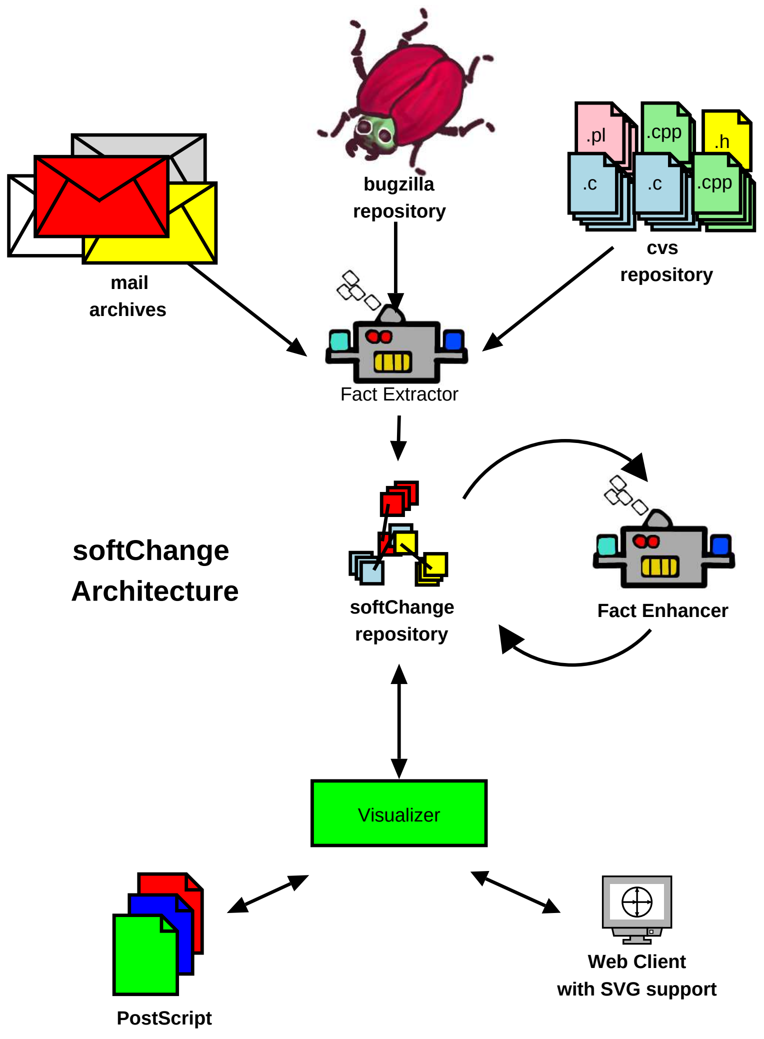

Figure 1: Architecture of Softchange

softChange

- What is softChange ?
  - Software Trails Repository - A relational database that stores all the historical data.
  - Software Trails Extractor - Extracts data from CVS, Changelogs, bug reports and emails.
  - Software Trails Analyzer / Fact Enhancer - Combines data in the repository to form MRs, and produce other useful statistics.
  - Visualizer - Visualize the data in the repository to aid the user in exploration and discovery.

Cheesegod

- What is Cheesegod?
  - Cheesegod is a system for testing invariants about change in a repository. Cheesegod models the repository as a graph and provides a query language.
  - `A(a,MR) { count(a.revisions) > 0 }`
  - $\forall a \in \mathbb{MR}(|\{x \in \mathbb{R}evision|((a,x) \in E)\}| > 0)$

MRs

- What is an MR?
  - Modification Request
  - Programmer submits a modification of the source code to the repository.
  - For CVS - when a programmer commits changes.

Files

- What is a File?
  - A point where a tree of revisions grows from
  - A filename
  - For CVS, the RCS file used to store revisions.

Revisions

- What is a Revision?
  - A Modification to a previous revision
  - The inital modification to a file
  - Revisions contain the changes, by combining revisions you can recreate the file for a particular version.

Authors

- What is an Author?
  - The user
  - The author is a the developer repsonsible for a revision in the repository.

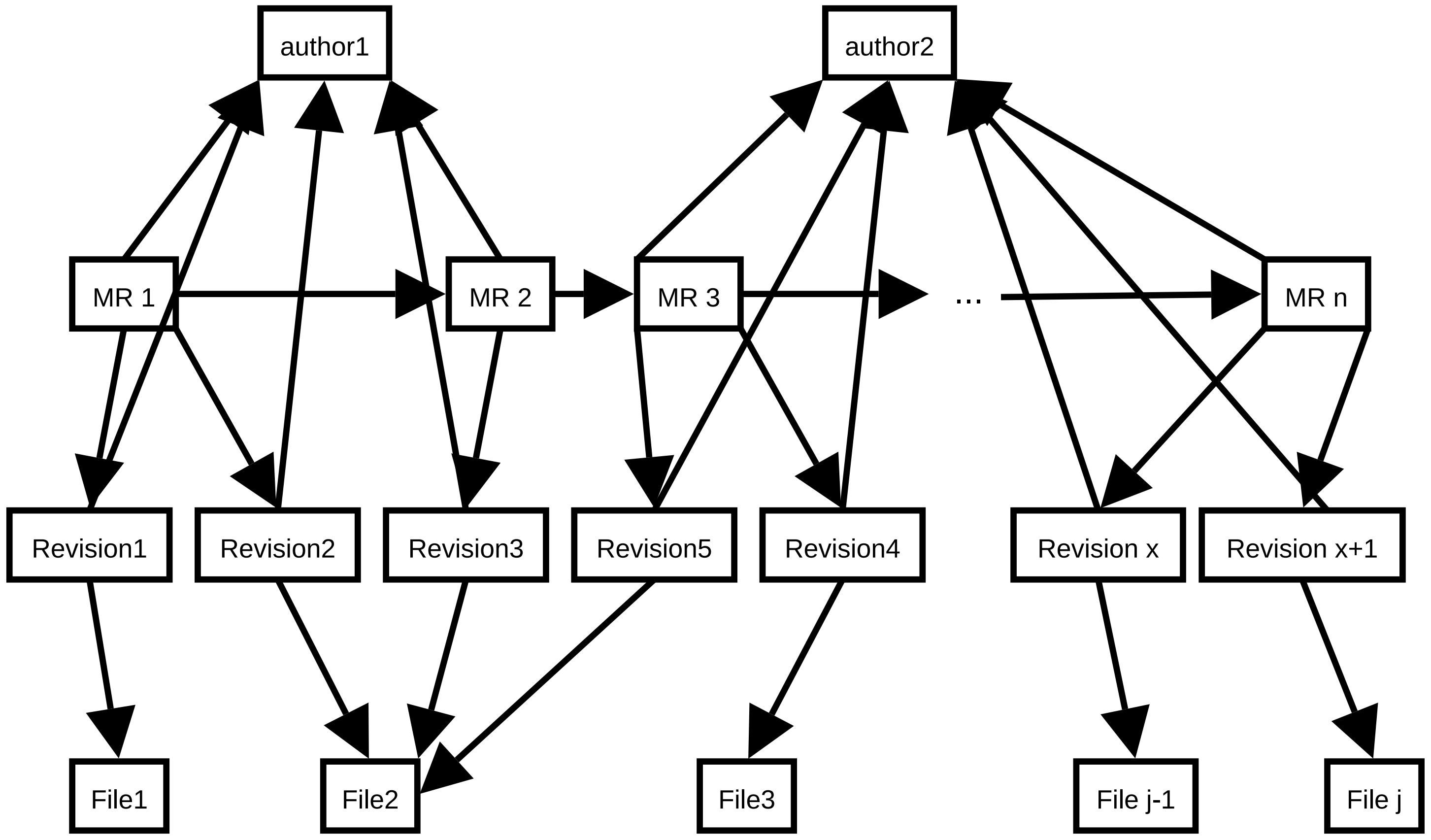

Figure 2: CheeseGod model of SCS

Immediate Metrics

- Metrics which are simply extracted data, with little or no elaboration
  - MRs are generated from Revisions in CVS.
  - Metrics are measurements
  - Metrics can be direct, indirect (derived) or predictive.

Immediate Metrics

- MRs have the following attributes
  - time
  - mrID
  - logEntry
  - authorname
  - authors
  - revisions
  - files

Immediate Metrics

- Revisions have the following attributes
  - time
  - revisionID
  - daterev
  - timerev
  - linesAdded
  - linesRemoved
  - diff
  - files
  - MRs
  - authors
  - Next Revision

Metrics of Change

- Previous Revision

Immediate Metrics

- Files have the following attributes
  - time
  - filename
  - module
  - directory
  - revisions
  - MRs

Immediate Metrics

- Authors have the following attributes
  - userid
  - name
  - revisions
  - MRs

softChange Metrics

- These are metrics immediately supported in softChange
  - MRs are generated from Revisions in CVS.
  - Generated from parsing source code.
  - Usefulness of certain metrics are debatable, but a metric can be used for different types of analysis like cluster analysis so extra information in the form of a metric isn't that bad.

softChange Metrics

- Revision Source Metrics through softChange
  - Change in Clean LOCS - LOC that are comment free
  - Number of Functions Added/Removed
  - Number of Methods Added/Removed
  - Number of Methods before/after
  - Number of Classes Added/Removed

softChange Metrics

- File based Metrics
  - File Coupling
  - File Author Coupling

softChange Metrics

- MR based Metrics
  - Files Added/removed
  - Methods Added/Removed
  - Diff Size
  - Clean Diff Size

softChange Metrics

- Author based Metrics
  - Author Coupling to a Module
  - Number of Files
  - Frequency of Modification

Cheesegod Metrics

- CheeseGod metrics are metrics which are either derived from the CheeseGod Model or Query Language.
- CheeseGod metrics are more relational than the softChange metrics as CheeseGod is working directly with a graph. Cheesegod metrics are also usually more global, a per project bases. (Many of these can be trivally answered in softChange as well).
- CheeseGod metrics shown can be used in both local metric (Anchor) and Global metrics (S,E,A,Sum,Std). Thus be aware if a metric is stated globally it can most likely be used locally.

Cheesegod Metrics

- Global Metrics
  - The Number of Revisions which are leaf nodes.

    ```
    count(S(a,Revision)
    {count(S(b,a.nextrev){true}) == 0 })
    ```
  -

Cheesegod Metrics

- Author based Metrics
  - Author $a only modifies files before another author `Anchor(a,Author,$a) { E(b,Author) { ( a != b ) -> A(i,a.revisions) { A(j,b.revisions) { before(i,j) } } }`
  - Author $a is a only has one commit `Anchor(a,Author,$a) { count(S(b,a.mrs){true}) == 1 })`
  - Max number of revisions per author `Max(a,Author) { count(S(b,a.revisions)true) }`
  - Avg number of revisions per author `Avg(a,Author) { count(S(b,a.revisions)true) }`
  - Min number of revisions per author `Min(a,Author) { count(S(b,a.revisions)true) }`

- Does there exist an author who for one or more files he is the only author

```
Anchor(a,Author,$a) { A(b,a.files) {
A(c,b.revisions) { eq(c.author,a.userid) } } }
```

Cheesegod Metrics

- Total Code Ownership
  - Does there exist an author such that for all of that author’s revisions, the files modified were files that the author originally created or had previously modified.

  -
    ```
    E(a,Author){
       A(b,a.revisions){ isFirstRevision(b) ||
          A(f,b.file){
            Abefore(r,f.revisions,b){ A(x,r.authors){ x == a
    ```

Cheesegod Metrics

- Subset Changes:
  - There exists an MR before that's subset of files is effectively a subset of files from a previous MR.

  -
    ```
    E(a,MR) { Ebefore(b,MR,a)
                 { a != b \&\& A(x,b.revisions)
                    { E(y,a.revisions) { eq(x.filename,y.filena
                    }
                 }
              }
    ```

Cheesegod Metrics

- Silent Authors:
  - For a project do all the authors submit MRs with no log comment?
  - `A(a,Author) { E(b,a.mrs) { length(b.log) < 1 } }`

Cheesegod Metrics

- Maintainer Author:
  - There exists an author who only modified files which were previously modified

  -
    ```
    E(a,Author) { A(b,a.revisions) {
    Ebefore(c,Revision,b) { neq(c.author,a.userid)
    && eq(b.filename,c.filename) } } }
    ```

Cheesegod Metrics

- Modular Author:
  - There exists an author who for all files they have modified, those files reside in the same directory.
  - `E(author,Author) { A(filea,author.files) { A(fileb,author.files) { eq(fileb.directory,filea.directory) } } }`

Cheesegod Metrics

- Recurrent Authors:
  - The Average Per Revision of all the revision authors modify the file before.
  - `Avg(a,Revision) { int( A(author,a.authors) { Ebefore(rev,author.revisions,a) { eq(a.filename,rev.filename) } }) }`

Cheesegod Metrics

- MR Deja Vu:
  - Average per MR over the average number of MRs before the current MR that have revisions of files in the same directory as the current MR.
  - `Avg(mr,MR) { count(S(a,MR) { before(a,mr) && A(f,a.files) { E(f2,mr.files) { eq(f.directory,f2.directory) } } }) / count(S(a,MR){ before(a,mr) }) }`

Cheesegod Metrics

- LOC Window:
  - The minimum, maximum and average number of lines added over a 30 day period.

  -
    ```
    Max(Revision1,Revision) {
    Sum(Revision3,S(Revision2,Revision) {
    abs(days(Revision2.daterev) -
    days(Revision1.daterev)) <= 30 }) {
    Sum(rev,Revision3.revisions) { rev.linesadd } }
    }
    ```

  -
    ```
    Avg(Revision1,Revision) {
    Sum(Revision3,S(Revision2,Revision) {
    abs(days(Revision2.daterev) -
    days(Revision1.daterev)) <= 30 }) {
    ```

```
  Sum(rev,Revision3.revisions) { rev.linesadd } }
  }
– Min(Revision1,Revision) {
  Sum(Revision3,S(Revision2,Revision) {
  abs(days(Revision2.daterev) -
  days(Revision1.daterev)) <= 30 }) {
  Sum(rev,Revision3.revisions) { rev.linesadd } }
  }
```

Cheesegod Metrics

- Modifying Author
  - The per revision average of the number of authors to modify a file after that revision.
  - `Avg(a,Revision) { count(S(b,Author) { neq(a.author,b.author) && Eafter(c,Revision,a) { eq(a.filename,c.filename) }})}`

Cheesegod Metrics

- Change to Stability
  - Does the MR $a modify a previously stable file? Given $d days.

  -
    ```
    Anchor(a,MR,\$a) {
            E(b,a.revisions) {
                    Abefore(c,Revisions,b) {
                            days(b.time) - days(a.time) > \$d
                    }
            }
    }
    ```

Cheesegod Metrics

- How Many Toes Are We Stepping on
  - Averaged per MR, how many unique authors were responsible for the revisions previous to the MR.
  - `Avg(m,MR) {`

```
                count(S(a,Author) {
                        E(r,m.revisions) {
                                A(r2,r.prevrev) {
                                        isAuthorOf(a,r2)
                                }
                        }
                })
```

```
        }
```

Cheesegod Metrics

- Coupled MRs
  - Percentage of MRs that span modules. A large percentage of MRs spanning multiple modules could indicate that the code is highly coupled. There has already been much research suggesting that the fact that source files were changed at the same time suggests they are possibly coupled.

  -
    ```
    count(
            S(a,MR) {
                    E(e,a.files) {
                            E(f,a.files) {
                                    neq(e.modules,f.modules)
                            }
                    }
            }
    ```

```
) /  count(MR)
```

Cheesegod Metrics

- Coupled Authors
  - Percentage of Author to Module coupling. How bound is an author to a module. Are most of their change across modules or do they focus heavily in a module. This is more so a measure of code ownership. Thus given a module and userid:
  -
```
count(S(f,File) {
        eq(f.module,"modulename") &&
        E(f.authors) {
                eq(a.userid,"userid")
        }
}) /
count(S(f,File) {
        E(f.authors) {
```

```
                        eq(a.userid,"userid")
                }
        })
```

Metrics Errata

- Other concerns for Metrics
  - Functions/Methods added created or deleted
    - Are there authors who don’t add/modify/delete functions.
    - Are there authors who add functions to one module or directory but only modify methods of other modules.
    - What authors exists that predominantly don’t modify functions.
  - Frequency Based Metrics
    - Interval based evaluation
    - Spectral Analysis (are there harmonics?)
  - Similarity of Changes
    - Are changes within the standard deviation of change?
    - What changes are not.

Observations

- In a repository of change we often can answer questions about the future related a change (the future which is already recorded in the repository).
- Predictor Metrics can easily be tested using the history in a repository.
- Did not cover using all of softChange 's features in the Cheesegod queries.
- An investigation into the deltas of common OO metrics would probably result in new kinds of metrics.
- Graph Metrics would also make sense. Especially if metrics are local. Our graph has a hierarchical quality so possibly metrics related to trees might be appropriate.

Summary

- We have proposed many metrics but many remain to be tested.
- The metrics we have proposed can be tested and generated using our tools and models (softChange and Cheesegod).
- Metrics can be evaluated on multiple levels: projectwide, intervals or locally.
- More research should be done on look at the deltas of current OO, and graph metrics.