Lecture 22
Knowledge Recovery and Software Reflexion Model
Today’s Agenda (I)

- Recap of Chianti
- Software Reflexion Model
Today’s Agenda (2)

• Discussion on application of software evolution research to development practices.
  • Information hiding principle
  • Concern graph
  • Delta debugging
  • Regression test selection
Recap of Chianti (1)

- Chianti is a dynamic change impact analysis tool.

1. Chianti analyzes differences between two versions as a set of atomic changes.

2. Chianti identifies a subset of regression tests that may change their behavior by identifying dynamic call graphs that include these changes. (Similar to RTS)

3. For each of those selected tests, Chianti identifies a subset of deltas that are responsible for behavior differences in those tests. (Similar to Isolation of fault-inducing changes)
Chianti Framework
First Phase

Pn → Program Differencing Tool
  => Identify Changes between Po and Pn

Po → T = \{t1, t2, ..tn\}
  Profiling Tool
  => Run T on Po

Pn → Delta
  (Dangerous Entities)

Po → Dynamic Call Graph

Pn → Affected Test Selection

T' \subset T
Chianti Framework
Second Phase

- Affected Test Selection
  - $T' \subset T$
  - $P_n$
  - Program Differencing Tool
    - Identify Changes between $P_0$ and $P_n$
  - $\Delta$

- Dynamic Call Graph
  - Profiling Tool
    - Run $T'$ on $P_n$

- Isolating Failure-Inducing Change
  - $D' \subset \Delta$

EE 382V Spring 2009 Software Evolution - Instructor Miryung Kim
Quiz
Software Reflexion Model

- Software Reflexion Models: Bridging the Gap between Design and Implementation, TSE 2001 (Extended Journal Version)
- Software Reflexion Models: Bridging the Gap between Source and High-Level Models. FSE 1995
Motivation

• What is this paper’s motivation?

• The drift between design and implementation happens during software evolution.
Research Problem

- Limitation of alternative existing approaches

1. Ignore the existing design document and rely on source code. => hard to understand source code (scalability) (initial investment on creating design doc does not pay off)

2. Rely on informal diagrams or design documents. => cannot have confidence / limited information inaccurate

3. Derive high-level models from source code. => cluttered,
Research Problem

- Limitation of alternative / existing approaches

1. Ignore the existing design document and rely on source code. => Source code or what reverse engineering tools would extract is overwhelming to programmers.

2. Rely on informal diagrams or design documents => Models are not always accurate.

3. Derive high-level models from source code => These may be different from what programmers expect to see.
Reflexion Model Approach

1. Enable a software engineer to produce a reasonable first-cut of a high-level model.

2. Enable him to map the high-level model and source code.

3. Then the reflexion model tool computes agreement and disagreement between the high-level model and the source code.
Step 1. Write a high-level model
Step 2. Extract a model from the program

- Use either a static analysis (source code) or a dynamic analysis (runtime execution).
- Call graph extraction
- Run time analysis (function calls, call sequences, event monitoring, etc.)
- e.g. Field, Rigi, Shrimp, etc.
Step 3. Define mappings between the high-level model and code.

```plaintext
[ file=*pager.*     mapTo=Pager ]
[ file=vm_map.*     mapTo=VirtAddressMaint ]
[ file=vm_fault\c   mapTo=KernelFaultHandler ]
[ dir=[un]fs        mapTo=FileSystem ]
[ dir=sparc/mem.*   mapTo=Memory ]
[ file=pmap.*       mapTo=HardwareTrans ]
[ file=vm_pageout\c mapTo=VMPolicy ]
```
Step 4.
Compare the models

Convergence: interactions expected by the developer
Divergence: interactions that were not expected by the developer
Absences: interactions that were expected but not found
Case Studies at Microsoft

- Subject Program: Microsoft Excel (over one million lines of C source code.)
- Task: a reengineering task
- Four week period
- The engineer found the approach valuable for understanding the structure and planning the reengineering effort.
Case Studies at Microsoft

- Subject Program: B. Griswold’s program restructuring tool (6000 lines C++ implementation)
- TasK: Design conformance -- which components do not adhere to layering principles?
- Divergences found by the reflexion model tool helped programmers revisit the locations and update the code to ensure the expected structure.
- There’s a similar study using SPIN OS as a subject program.
Discussion

- When will you use it?
  - Check what you intended matches your source code
  - Working design document => program understanding
  - When not to use this - small program just read it
- What do you like about it?
  - Iterative design conformance checking
• What are limitations of reflexion model?
  • mapping is painful.
  • mapping is only restricted to entities
  • high level models only captures structural aspects. (types, temporal semantics)
  • crosscutting concerns  --> many high level models that model different aspects
Contributions

- **Lightweight** -- minimal burden on a programmer side
- **Approximate** -- can start with a coarse model and then refine it iteratively.
- **Scalable** -- can run a million lines of code
My general thoughts on Software Reflexion Model

- Software Reflexion Model allows programmers to check design conformance to a high-level mental model.
- A very simple idea, *yet very powerful, and it has practical impact*
- It bridges the gap between software architecture (design) models and implementation models
- Its use as a design conformance tool is somewhat similar to program verification.
Practical Implications of Software Evolution Research

- Concern Graph
- Delta Debugging
- Regression Testing Selection
Preview for Next Monday

- We will continue to discuss reverse engineering and knowledge discovery => software metrics & visualization
- Lanza et al. Polymetric Views (Mon, 4/20)
Announcement

- Preliminary grading guidelines for projects / literature surveys are uploaded on the blackboard.
- There is no class lecture on 29th. Use it for your project presentation / report preparation.