Announcements

- Fill out Faculty Course Evaluations
  - Web site closes Friday
- Additional Blackboard survey to be posted
  - Feedback on improvements for next year

Reengineering with Reflexion Models: A Case Study

IEEE Computer, Gail Murphy
and David Notkin

17-654/17-765
Analysis of Software Artifacts
Jonathan Aldrich

Task

- Reengineer Excel code
  - 1.2 million LOC
  - Extract components

The Challenge

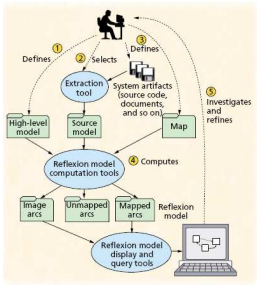
- Gain knowledge to perform reengineering
- Typical strategy: sketch a model
  - Risk: model may not correspond to code
- System goal: build a validated model
  - Task-specific modeling
  - Lightweight for early feedback on model
  - Iterative to allow refinement of model

Previous Techniques

- Automated approaches
  - Automatically construct model from source
  - Interactions are hard-coded
    - May be inappropriate for the task
  - Granularity fixed
    - Enough detail?
    - Too much detail?
- Semi-automated approaches
  - Allow user to cluster low-level source code components in customized way
  - Tough to scale to larger systems

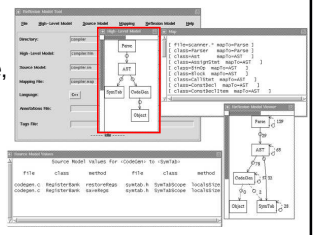
Basic Approach

- Hypothesize a Model
- Describe mapping to code
  - Can use tools customized to task
- Validate model vs. code
  - Tool shows differences
- Refine model and/or mapping and iterate



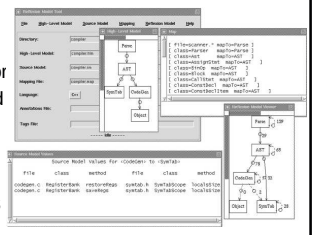
Defining a Model

- Graph of nodes and arcs
- Based on knowledge, documentation, or browsing code
- Can be arbitrary
- Estimate: 15-60 min



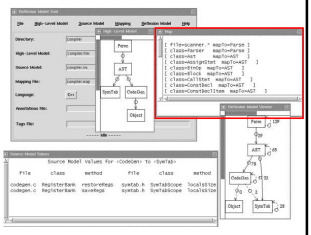
Extract Source Model

- Use tool
- Excel example
  - Call-graph constructor
  - Approximates desired dataflow information
  - Could be different in other applications
- Shows dependences between source elements



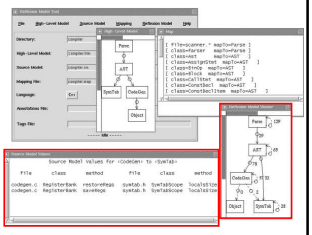
Defining a Mapping

- Source elements to nodes
  - Files
  - Classes
  - Functions
- First cut may be approximate
- Estimate: 10-30 min



Reflexion Model

- Shows high-level model
  - Convergences
  - Divergences
  - Absences
- Can investigate arcs
  - Provides valuable information for refining model



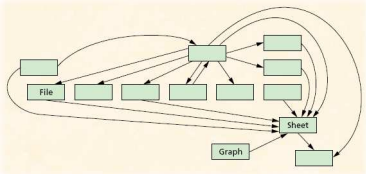
Refinement Process

- Address divergences/absences
  - Modify model
  - Modify mapping
    - e.g., function g belongs in file f, but was in file p instead
    - Tendency to add functions where the cursor is!
- Refine model
  - Split a node into parts, specify substructure

State of Excel Documentation

Excel Internals . . . explains the philosophy of a few of the basic things in Excel, like the cell table formulas, memory allocation, a little bit about the layer [a special interface with the operating system that allows Microsoft to use the same Excel core on both Windows and Macintosh platforms]. . . . It's very sparse. We don't necessarily rely on that for people to learn things. I'd say we have a strong oral tradition, and the idea is that the mentor teaches people or people learn it themselves by reading code. . . . Over the course of a project, it goes from mostly truthful to less truthful, and then we have to fix it up. We don't fix it up as we go along on a project. We will give it some attention between projects.

Initial Modeling Process



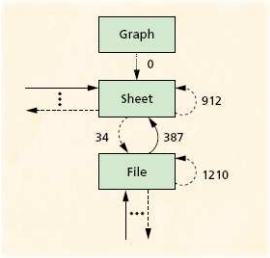
- Reading Excel Internals
- Brief discussion with team members
- Drew “natural” model

Source Model and Mapping

- Source Model constructed by internal Microsoft call-graph building tool
  - 77,746 calls!
- Mapping
  - 170 lines long
  - Describes 400 files
  - Took a few hours

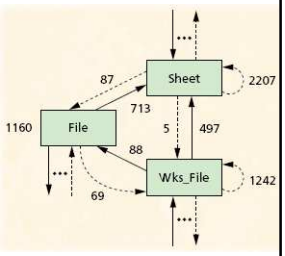
Initial Reflexion Model

- Tasks
  - Update model
    - if reasonable interactions missing
  - Investigate edges
    - to learn about source
  - Update map
    - exceptions for functions logically in another module
    - ultimately 1000 lines long
  - Extended source model
    - global variables
- Detailed focus on relevant parts of system
- Work done with scripts



Resulting Model

- Benefits
  - Understanding of system
    - Unexpected dependences
  - Feasibility of reengineering
    - How many arcs would be cut?
  - Aid in component isolation
    - Insert conditional compilation based on map



Reengineering Tool: Lessons Learned

- Task-specific views are important
  - Developer didn’t want to waste time on irrelevant parts of the system
- Connection to code important
  - Both for understanding and for reengineering task itself
- Both text and GUI interfaces needed
  - Most real work done with text!
- Adaptable tools needed
  - Engineer wrote scripts to process input/output files

Course Summary

Topics

- Program analysis
- Soundness, Precision
- Analysis tools
  - Fluid
  - PREfix
  - Metal
  - Fugue
  - Daikon
- Model checking
- Testing
  - Prioritization
  - Coverage
  - Generation
- Defect Prediction
- Reverse engineering and re-engineering
- Security
  - Design: attack graphs
  - Code: privilege separation
  - Timing attacks
- Next year
  - Theorem proving
  - Performance analysis
  - Reliability analysis

Themes

- Tradeoffs among analysis approaches
  - Static vs. dynamic
  - Automated vs. manual
- Soundness and precision
  - False negatives, false positives
- Practical considerations
  - Focus on task
  - Scaleability
  - Incrementality
- Breadth of analysis
  - Correctness, Security, Dependability, Understanding
  - Design, Code, Maintenance

Experience

- Program analysis
- Model checking
- Tradeoffs among analysis
- Tools

- Next year
  - Shorter, more focused assignments
  - Experience wider variety of tools

Takeaways

- Knowledge of how tools can help
- What's out there (now and in the near future)
  - Experience writing and using analyses
- Understanding resource allocation
  - Different analyses for different goals, at different points in the life-cycle
- Ability to evaluate new analysis techniques
  - What is the technique giving you
    - Assurance? Bug finding?
  - Ask hard questions about practicality
    - Incremental? Scaleable? Effort? Task-specific?