Scalability Solutions for Program Comprehension through Dynamic Analysis

Andy Zaidman

Philips Research, Eindhoven
15th December 2006
Outline

• Where I come from
• What I did
• Where we are going
Where I come from…

• 2002-2006: working on “ARRIBA” project
  – Architectural Resources for the Restructuring and Integration of Business Applications
  – IWT project, category: “Generisch Basis-Onderzoek Universiteiten”

• 9/2006: PhD University of Antwerp
  – Lab On Reengineering, prof. dr. Serge Demeyer

• 10/2006: TU Delft
ARRIBA

• Project theme
  – Use aspects in legacy systems (C, Cobol) to extract knowledge
  – University of Antwerp responsible for *dynamic analysis*

• Partners
  – Main *academic* partners: Universiteit Antwerpen, Vrije Universiteit Brussel, Universiteit Gent
    • University of Bern, Université Catholique de Louvain
  – *Industrial* partners: KBC, KAVA, Christelijke Mutualiteit, Banksys, Anubex, Toyota Europe, Inno.com
Context of what I did

• **Program comprehension**
  – Gather knowledge *efficiently* about (unknown) project
  – Cognitive scalability: don’t present too much information to user

• **Dynamic analysis**
  – Run-time information from execution scenario
    + Takes into account polymorphism
    + Precision of knowledge w.r.t. execution scenario
  – Large traces, overhead

→ *scalability* (cognitive & technical)
It’s time for a fairytale…

• Who doesn’t know the story of Hans & Gretel…
  – Their father was a lumberjack and the family was very poor and didn’t have enough food to feed the whole family…
  – … therefore, Hans & Gretel’s parents decided to abandon their children in the forest.
  – Luckily, Hans & Gretel are very smart kids… so they devise a plan to get back home!
Plan A: use a map!

- Cartography
  - Maps have existed for thousands of years, even before writing existed
  - Making an accurate map is time intensive & very expensive

- “Hans & Grietje” bicycle route in Barneveld (between Arnhem & Amersfoort)
Plan B: They can leave clues behind!

- At specific points (e.g. crossroads), leave small rocks or bread crumbs behind → follow the same path back home!
The moral of the story

• *Don’t end up at the confectionary house (the house with the witch)!*

• Maps are too expensive!

• Use little rocks that you leave behind at key points during your walk

• Use these to “backtrack” your way home
Back to reality…

- There is parallel…
  - Using a map can be compared to using UML class diagrams
  - Using little rocks as clues can be compared to using an execution trace of an application
- This research is about exploiting these clues!
  → Dynamic analysis
- However, sometimes, there are too many of these clues
  - How to abstract? How to guide the end-user?
...what we propose

- A way to identify *need-to-be-understood* classes or components in a large system
  - … a first indication as to where to start initial program comprehension efforts
  - … should be complemented by using other, more in-depth program comprehension techniques or by manual code-inspections
  - “Key Concept Identification”
Overview of the process

1. Define execution scenario
   - Implies end-user knowledge of how to run the application
2. Trace the application
3. Apply technique
4. Interpret results
Coupling based

• High level of coupling can indicate classes that have *control* over the execution ("influential classes")

• Add notion of *indirect coupling*, because classes that work tightly together with influential classes are also important and need-to-be-understood

• We use the HITS webmining algorithm to add the concept of indirect coupling.
HITS webmining

- Each node in the graph gets two values associated with it: hubiness and authority
- Recursive definition:
  \[ h_i = \sum_{i\rightarrow j} a_j \]
  \[ a_j = \sum_{i\rightarrow j} h_i \]
- Start with hub and authority equal to 1.
- Iteratively update the weights.
- Algorithm converges to stable set of weights*.

* See Kleinberg, Journal on ACM, 1999 (5)
Webmining
Webmining

Initialize weights to 1

Diagram showing a network of nodes with weights (1,1).
Webmining

Recursively update the weights

\((2, 0)\) \(\rightarrow\) \((1, 2)\)

\((3, 0)\) \(\rightarrow\) \((0, 3)\)

\((0, 1)\)
Webmining

Recursively update the weights
Webmining

Recursively update the weights

(11,0)  
(14,0)  
(6,11)  
(0,11)  
(0,6)
Webmining

Hubs

Authorities
Result of webmining

• We specifically look at the hubiness scores for each class/module
• Results: ranking of classes/modules according to relative “importance”
• Take the highest 15% ranked classes/modules and consider these for program comprehension purposes
Research approach

• 2 open source cases (Ant & JMeter)
  – Validation with in-depth documentation
• 1 industrial case
  – Validation with developers/maintainers
Open source: Ant & JMeter

- Results from Ant:
  - Recall: 90%
  - Precision: 60%

- JMeter very similar
Industrial case

- KAVA = Koninklijke Apothekersvereniging Van Antwerpen
- Active in healthcare insurance business
- Kava acts as a financial go-between
  - Pharmacists
  - Health insurance institutions
- C system dating from around 1990
  - non ANSI-C, UnixWare C
Aspects and C

• We used *Aspicere* developed @ University of Ghent (Bram Adams, Kris De Schutter)
• Integrating Aspicere in a legacy environment troublesome
  – non-ANSI C constructs (e.g. delayed parameter typing)
  – Makefiles heterogenously structured
    • Not only calls to GCC, but also ESQL, scripts, …
## Kava results

<table>
<thead>
<tr>
<th>Module</th>
<th>Aut</th>
<th>Hub</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ica/project/deelproject/TDFS/sources/e_tdfs_mut1.c</td>
<td>0.915478</td>
<td>0.814941</td>
</tr>
<tr>
<td>/ica/project/deelproject/TDFS/sources/tdfs_mut1_form.c</td>
<td>0.872067</td>
<td>0.45397</td>
</tr>
<tr>
<td>/ica/project/deelproject/TDFS/sources/tdfs_bord.c</td>
<td>0.0</td>
<td>0.397726</td>
</tr>
<tr>
<td>/ica/project/deelproject/TDFS/sources/tdfs_mut2.c</td>
<td>0.594401</td>
<td>0.164278</td>
</tr>
<tr>
<td>/ica/project/algemeen/Show_listing_box/tools.c</td>
<td>0.198554</td>
<td>0.164278</td>
</tr>
<tr>
<td>/ica/project/deelproject/batch/PROCESSOR/RELEASE/io.c</td>
<td>0.716924</td>
<td>0.12548</td>
</tr>
<tr>
<td>/ica/schermen/cprogs/csrout.c</td>
<td>0.198554</td>
<td>0.0321257</td>
</tr>
<tr>
<td>/ica/project/algemeen/apoteek/tarpargeg/tarpargeg.c</td>
<td>0.55099</td>
<td>0</td>
</tr>
<tr>
<td>/ica/schermen/cprogs/csrdtl.c</td>
<td>0.213674</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/algemeen/strcpy/UW_strncpy.c</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/algemeen/Show_listing_box/td.ec</td>
<td>0.198554</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/deelproject/batch/PROCESSOR/RELEASE/cache.c</td>
<td>0.716924</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/algemeen/decfties/decfties.c</td>
<td>0.766083</td>
<td>0</td>
</tr>
<tr>
<td>/ica/project/deelproject/tapes/sources/weglf.c</td>
<td>0.198554</td>
<td>0</td>
</tr>
<tr>
<td>/ica/schermen/cprogs/get_request.c</td>
<td>0.198554</td>
<td>0</td>
</tr>
</tbody>
</table>
Feedback on results

• We interviewed two Kava developers before showing them our result set.
• Question: which module(s) is/are most important?
  – D1 mentioned e_tdfs_mut1.c and tdfs_mut2.c
  – D2 mentioned e_tdfs_mut1.c
• Our result set ranks them in the top 4, with e_tdfs_mut1.c as the most important.
• Counter indication: filenames contain “tdfs”.

# Kava effort analysis

<table>
<thead>
<tr>
<th></th>
<th>normal</th>
<th>with AOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build cycle</td>
<td>15 → 20 min</td>
<td>17h38m</td>
</tr>
<tr>
<td>Execute scenario</td>
<td>1h30m</td>
<td>+- 7h</td>
</tr>
<tr>
<td>Logfile size</td>
<td>90 GB (600 MB compressed)</td>
<td></td>
</tr>
<tr>
<td>Webmining analysis</td>
<td>10h</td>
<td></td>
</tr>
</tbody>
</table>

Granted, prototype status of tools, but still... this is *not good!*
Conclusion

• Our approach allows us to:
  + Recall 90% of the classes
  + Precision of 60%
  – What about the effort? Acceptable?
  – Is there always a good execution scenario available?
Pointers

• Applying Webmining Techniques to Execution Traces to Support the Program Comprehension Process

• Regaining Lost Knowledge through Dynamic Analysis and Aspect Orientation - An Industrial Experience Report
1. Help extend Bas’ work on visualizing test suites
   • See his recent publication:
     Visualizing Testsuites to Aid in Software Understanding
     Conference on Software Maintenance and Reengineering,
     Amsterdam, March 2007

2. Short-term future (first half of 2007)
   • Investigate how developers test (see next slides)

3. Mid-term future (until approx. 10/2007)
   • Extend research on datamining through e.g.
     “sequential pattern mining”
Testing behavior

- Correlate when production code and test code are committed.
- Mine version control systems
  - e.g. CVS, Subversion, Clearcase, …
- Early results from open source cases available (all use Subversion):
  - Apache Ant
  - Apache Tomcat
  - JBoss
Ant testing history
Time for some Q&A

Some questions from us to you:

1. Are you interested in a “Tour d’horizon”: talk to you guys and see what problems you have that can be alleviated with the tools that we are currently developing.

2. Would you like to participate in the “test mining” experiment?