Software Model Mining:
static analysis, passive learning, active learning

Dennis Dams, TNO-ESI
AIS Colloquium Thu 2016.09.29
MOTIVATION & INTRO
Motivation

Legacy code: large software systems that we don’t know how to cope with but that are vital to our organization

Model Driven Software Engineering

Using modeling languages that are

• high level
• supported by rich tools
• domain specific
Motivation

Software Model Mining

Lagging

Leading
3 Selected Technologies for Software Model Mining
3 Selected Technologies for Software Model Mining

- **glass box analysis**
  - source code analysis / static analysis

- **active learning**

- **passive learning**
  - execution analysis / dynamic analysis
TODAY → TOMORROW
## What To Expect? – “Tomorrow”

<table>
<thead>
<tr>
<th>Type of use case</th>
<th>Objective of the model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract <em>specification</em> models</td>
<td>prescriptive (specification model)</td>
</tr>
<tr>
<td>Extract models for <em>analysis</em>, <em>simulation</em>, <em>verification</em>, …</td>
<td>descriptive (aspect model)</td>
</tr>
<tr>
<td>Extract models that support <em>health monitoring</em>, <em>testing</em>, …</td>
<td>supportive</td>
</tr>
</tbody>
</table>
Specification models: where they usually are

- bug fixes
- new features
- ...

“Legacy Code”
Specification models: where they should be

Model-Driven Software Engineering

- corrections
- new features
- ...
## What’s Already There – “Today”

<table>
<thead>
<tr>
<th>Tools</th>
<th>General findings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>code analysis</strong></td>
<td></td>
</tr>
<tr>
<td>Xtext &amp; qvto</td>
<td>• size and complexity of code may be limiting factor</td>
</tr>
<tr>
<td>Rascal</td>
<td>• expect to use in combination with dynamic approaches</td>
</tr>
<tr>
<td>GCC, Clang, Eclipse CDT</td>
<td></td>
</tr>
<tr>
<td><strong>passive learning</strong></td>
<td></td>
</tr>
<tr>
<td>Disco</td>
<td>• preprocessing crucial for getting informative output</td>
</tr>
<tr>
<td>ProM</td>
<td>• requires interaction with domain expert</td>
</tr>
<tr>
<td>Mint</td>
<td>• cannot learn more than what’s present in the log traces</td>
</tr>
<tr>
<td>DFASat</td>
<td>• suitable for learning about the ways a component is used (environment)</td>
</tr>
<tr>
<td><strong>active learning</strong></td>
<td></td>
</tr>
<tr>
<td>LearnLib</td>
<td>• suitable for learning the complete internal behavior of a component</td>
</tr>
<tr>
<td></td>
<td>• additional research identified, required to extend applicability (data params, nondet.)</td>
</tr>
</tbody>
</table>

survey article: *Automated API Property Inference Techniques* by Robillard et. al.
### Getting From Today to Tomorrow

<table>
<thead>
<tr>
<th>ESI activities (various projects)</th>
<th>Academic Partnering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>code analysis</strong></td>
<td>(Renaissance project with FEI: “COM cleanup”)</td>
</tr>
<tr>
<td><strong>passive learning</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>SWMM Workbench</strong></td>
<td></td>
</tr>
<tr>
<td>• more use cases</td>
<td></td>
</tr>
<tr>
<td>• data collection operators</td>
<td></td>
</tr>
<tr>
<td>• more preprocessing</td>
<td></td>
</tr>
<tr>
<td>• online/streaming</td>
<td></td>
</tr>
<tr>
<td><strong>active learning</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Mini-roadmap:</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>phase 1</strong>: reconstruct models from generated components</td>
<td></td>
</tr>
<tr>
<td>• <strong>phase 2</strong>: same for legacy components</td>
<td></td>
</tr>
</tbody>
</table>
Passive Learning – Software Model Mining Workbench
Active Learning – Phase 2

(no model)

legacy implementation

model

new implementation

active learning

validate/cleanup

generate
CASE STUDY: LEARNING AN INTERFACE MODEL
Obtaining Traces as Input for Passive Model Learning

• Observation: execution traces are difficult to obtain
  – practical difficulties
    • hard to find traces
    • hard to understand how traces were generated
  – technical difficulties
    • not clear how many traces is enough for our purposes

• Workaround solution: produce traces ourselves
  – based on Promela* model of the interface model, specifically constructed for this purpose
    (*Promela is the input language of the Spin model checker)
Seeking Expert Advice

- Sicco Verwer (TU Delft)
- tool: DFASat – winner of StaMinA tool competition

- Attempt 1: based on traces that we generated and sent to Sicco
- Attempt 2: based on traces that Sicco generated specifically for input to his DFASat tool (knowing precisely how the tools works), using the Mealy machine (equivalent to the Promela model) that we sent him
Attempt 2 – based on traces generated by Sicco
From Passive to Active Learning

finite state model of behavior

active learning

passive learning

execute and log

input

output

software component

traces
A Web Interface to the Interface Model

```
dennis@dennis-VirtualBox:~$ wget -q -O - http://drd.pythonanywhere.com/actlearn/ex8_data/_ins

-dennis@dennis-VirtualBox:~$ wget -q -O - http://drd.pythonanywhere.com/actlearn/ex8_data/_outs
```
Active Learning On Interface Model

[Diagram showing a model with nodes and arrows labeled with conditions like 'al2/none', 'sp1/none', etc.]
THANK YOU