Behavioral Dependency Measurement for Change-proneness Prediction in UML 2.0 Design Models

Ah-Rim Han

Department of Computer Science, Korea Advanced Institute of Science and Technology

COMPSAC 2008, Turku, Finland
July 29, 2008
Contents

- Introduction
- Goal of Our Research
- Change-proneness Prediction
- Overview of Our Approach
- Behavioral Dependency Measurement
- Case Study
- Related Work
- Conclusion and Future Work
Software changes either to enhance the functionality or to fix bugs

Some part of the software may be more prone to be changed than others

Identifying the parts which are more prone to be changed, *change-proneness*, can be helpful

- Ex) Re-design the classes which are sensitive to change in OO
Several research efforts for predicting change-prone classes have been made on *source codes*

What if change-prone classes can be predicted earlier phase in the SDLC…?
Benefit of model-based change-proneness prediction

- Constructing a flexible and stable software would be much easier
  - By modifying the current design before implementing to codes
  - By making a decision among candidate design models

- Development cost would be reduced
  - Largest percentage of software development effort is spent on rework and maintenance
Goal of Our Research

- Provide the Behavioral Dependency Measure (BDM) for change-proneness prediction
  - Based on UML 2.0 design models
    - Sequence diagram (SD), Class diagram (CD) and Interaction overview diagram (IOD)
  - Based on behavioral dependencies of pairs of objects
Assumption

- Changes occur by change propagation

- Changes can be predicted by examining dependencies of pairs of objects

- When an object sends a message to the other object, modifying the object receiving the message may affect the object sending the message

- High intensity of a dependency represents high possibility of changes to be occurred
Definition

- An object sending a message has a **behavioral dependency** to the object receiving the message
  - Direct behavioral dependency
  - Indirect behavioral dependency

Sequence Diagrams (SDs)
Strategies for accurate prediction

Execution rate of a message

- Probabilistic aspect
  - Branch control structure (alt combined fragment in a SD)

- Expected aspect
  - Operational profile in the IOD
Strategies for accurate prediction (Cont’d)

- Inheritance and polymorphism
Overview of Our Approach

Model-based change-proneness prediction

Behavioral Dependency Measurement

- Deriving all the Reachable Paths for all object-pair
- Synthesizing OBDGs into Object System Behavioral Dependency Graph
- Constructing Object Behavioral Dependency Graph for each SD
- Calculating Behavioral Dependency Measure for every class in the system
- Summing the number of weighted Reachable Paths for all class-pair
- Sum of the weighted reachable paths
- Calculating Behavioral Dependency Measure for every class in the system

Predicting change-proneness
OBDG \textsubscript{A} = \{O,M\}, where
- O: objects in the SD A
- M: \(m_n(m_b,m_{meL},m_{meH})\)
  - \(m_n\): message name
  - \(m_b\): instance of a backward navigable message
  - \(m_{meL}\): probabilistic execution rate in SD
  - \(m_{meH}\): expected message execution rate in IOD
Synthesizing OBDGs into OSBDG

Constructing OBDG and OSBDG (2/2)
Where we are now

Model-based change-proneness prediction

Behavioral Dependency Measurement

- Deriving all the Reachable Paths for all object-pair
- Synthesizing OBDGs into Object System Behavioral Dependency Graph
- Constructing Object Behavioral Dependency Graph for each SD
- Summing the number of weighted Reachable Paths for all class-pair
- Sum of the weighted reachable paths
- Calculating Behavioral Dependency Measure for every class in the system
- Predicting change-proneness
Deriving all reachable paths for all pair of objects in the system using OSBDG

An example of reachable path set from o1 and o3: \{ab, b'\}
Where we are now

Model-based change-proneness prediction

Behaviorsal Dependency Measurement

- Deriving all the Reachable Paths for all object-pair
- Synthesizing OBDGs into Object System Behavioral Dependency Graph
- Constructing Object Behavioral Dependency Graph for each SD
- Predicting change-proneness

Reachable Path Set

Summing the number of weighted Reachable Paths for all class-pair

Sum of the weighted reachable paths

Calculating Behavioral Dependency Measure for every class in the system

BDMs

OSBDG

OBDGs

CD

SD

IOD
Calculating BDM (1/2)

- Summing the number of weighted reachable paths for all pairs of classes in the system

$$\text{SumWRP}(c_1, c_2) = \sum_{s \in \text{RPS}(o_1, o_2)} DF(s) \times f_{meH} \times f_{meL}$$

Since $\text{RPS}(o_2, o_4) = \{bc, d\}$, $\text{SumWRP}(c_1, c_2) = \left( \frac{1}{2} \times 0.8 \times 1 \right) + (1 \times 0.8 \times 1)$

- $DF(s) = \frac{1}{d}$
- $d$: distance length which is the number of messages in the corresponding reachable paths
Calculating BDM (2/2)

Calculating BDM for every class in the system

\[ BDM(c) = \sum_{c_n \in C} SumWRP(c1, c_n) \]

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
<th>c6</th>
<th>BDM(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>0</td>
<td>0.9</td>
<td>0.5</td>
<td>0.67</td>
<td>0</td>
<td>0.2</td>
<td>2.27</td>
</tr>
<tr>
<td>c2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>0.8</td>
<td>0</td>
<td>2.8</td>
</tr>
<tr>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>c4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Therefore, the class c2 is likely to be changed most in the system.
Goal

To show that BDM is the useful and additional explanatory variable for change-proneness prediction

Experiment Design

Make two multivariate regression models with different independent variable set
• Only C&K metrics* vs. BDM in addition to C&K metrics
• Compare goodness of the fit of those models

Studied Environment

Input model: JFreeChart
- Is an open-source Java class library for generating various types of charts
- Reversed codes (ver. 1.0.0) into UML models
  - IOD is not applicable
  - CD is generated from RSA 7.0
  - SD is constructed based on successive synchronous calls (i.e., reachable paths)

SD reversed from the successive synchronous calls existed in JFreeChart source codes
Case Study (3/4)

Studied Environment (Cont’d)

- **Tool: BADAMO (BehAvioral Dependency Analyzer of UML MOdels)**
  - Calculates the BDM
  - Is implemented based on EMF (Eclipse Modeling Framework)
  - Imports UML 2.0 models in the format of XMI generated from RSA (Rational Software Architect) 7.0

- **Method for building prediction model: stepwise multiple regression**
  - Dependent variable: change-proneness
    - Total amount of changes (source lines of code added and deleted) across the six releases (v.1.0.1 ~ 1.0.6)
  - Independent variables
    - C&K metrics (NOC, DIT, WMC, RFC, CBO, LCOM) and BDM
Case Study (4/4)

_results

- **Models with only C&K metrics**
  - Explains around 56% (adjusted \( R^2 \) of 0.55)
  - Selected variables
    - WMC (1\textsuperscript{st}), CBO (2\textsuperscript{nd}), and LCOM (3\textsuperscript{rd})

- **Model with BDM in addition to C&K metrics**
  - Explains around 64% (adjusted \( R^2 \) of 0.64)
  - Selected variables
    - WMC (1\textsuperscript{st}), BDM (2\textsuperscript{nd}), CBO (3\textsuperscript{rd}), LCOM (4\textsuperscript{th}), and NOC (5\textsuperscript{th})

\( R^2 \) is increased by 9 percent or 20 percent of the unexplained variance using BDM.
Conclusion

- Model-based change-proneness prediction using BDM
  - Help to redesign the change-prone classes easily
    - Make a stable software
    - Reduce the development cost of software
  - Can be used to visualize the problematic spots
    - Improve understandability of software
Future Work

- Extend BDM to take into account other dependencies
  - Time, etc.
- Investigate other applications of BDM
  - Fault-proneness prediction, object allocation in a distributed system, etc.
- Visualize change-prone classes on the modeling tools
  - Rational Rose, ArgoUML, RSA (Rational Software Architect), etc.
Thank You.