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

KAIST SE LAB
KAIST Software Engineering Laboratory

Contents

Introduction

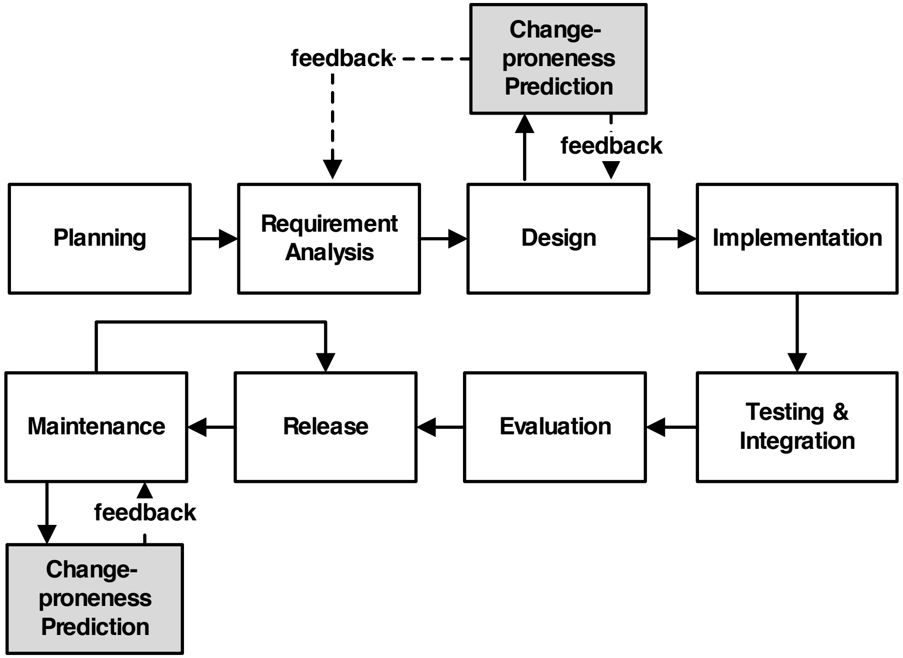
- Software changes either to enhance the functionality or to fix bugs

- 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

Motivation (1/2)

- Several research efforts for predicting change-prone classes have been made on *source codes*



Change-proneness prediction in Software Development Life Cycle (SDLC)

- What if change-prone classes can be predicted earlier phase in the SDLC…?

Motivation (2/2)

- 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

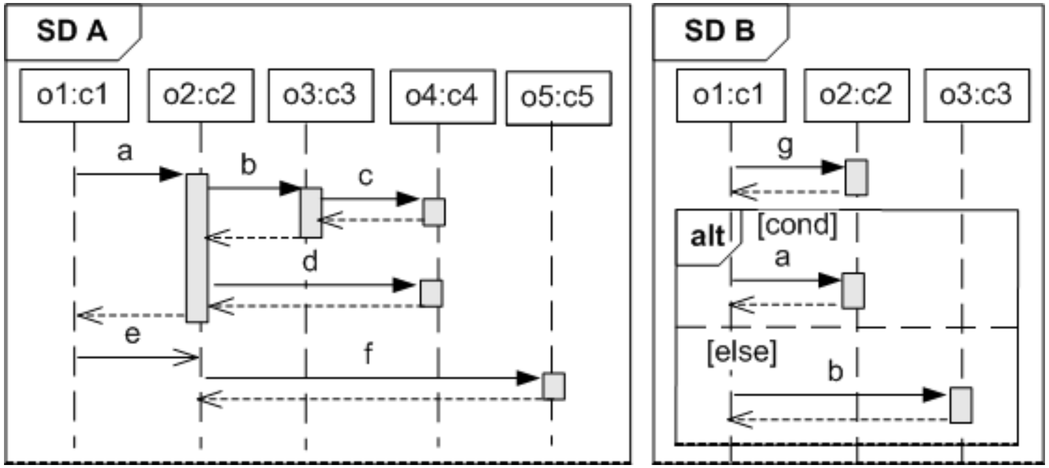
Change-proneness Prediction (1/4)

- 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

Change-proneness Prediction (2/4)

Definition

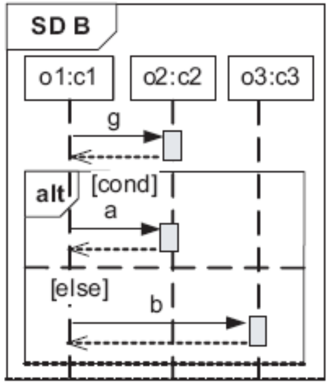
- An object sending a message has a behavioral dependency to the object receiving the message
  - Direct behavioral dependency
  - Indirect behavioral dependency



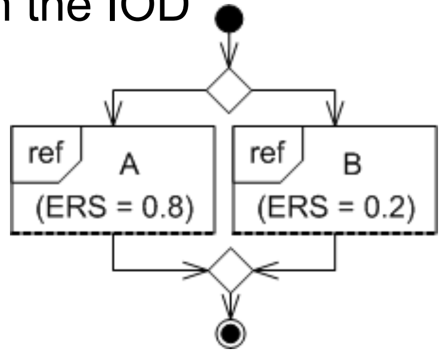
Sequence Diagrams (SDs)

Change-proneness Prediction (3/4)

- Strategies for accurate prediction
  - Execution rate of a message
    - Probabilistic aspect
      - Branch control structure (*alt* combined fragment in a SD)

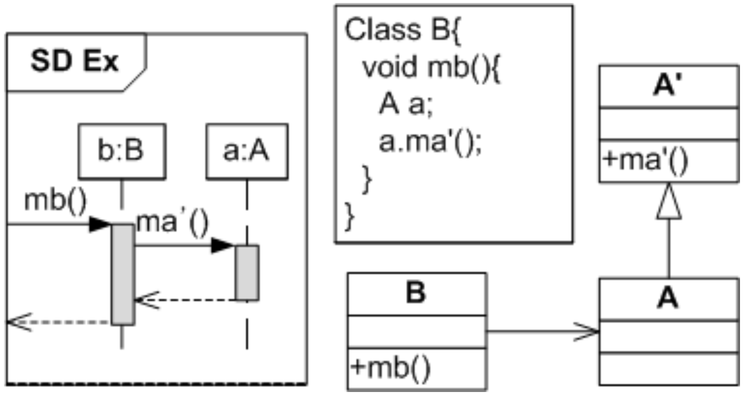


    - Expected aspect
      - Operational profile in the IOD



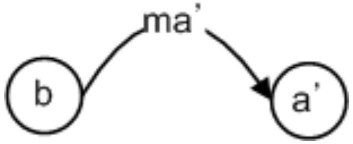
Change-proneness Prediction (4/4)

- Strategies for accurate prediction (Cont’d)
  - Inheritance and polymorphism



```
Class B{
  void mb(){
    A a;
    a.ma'();
  }
}
```

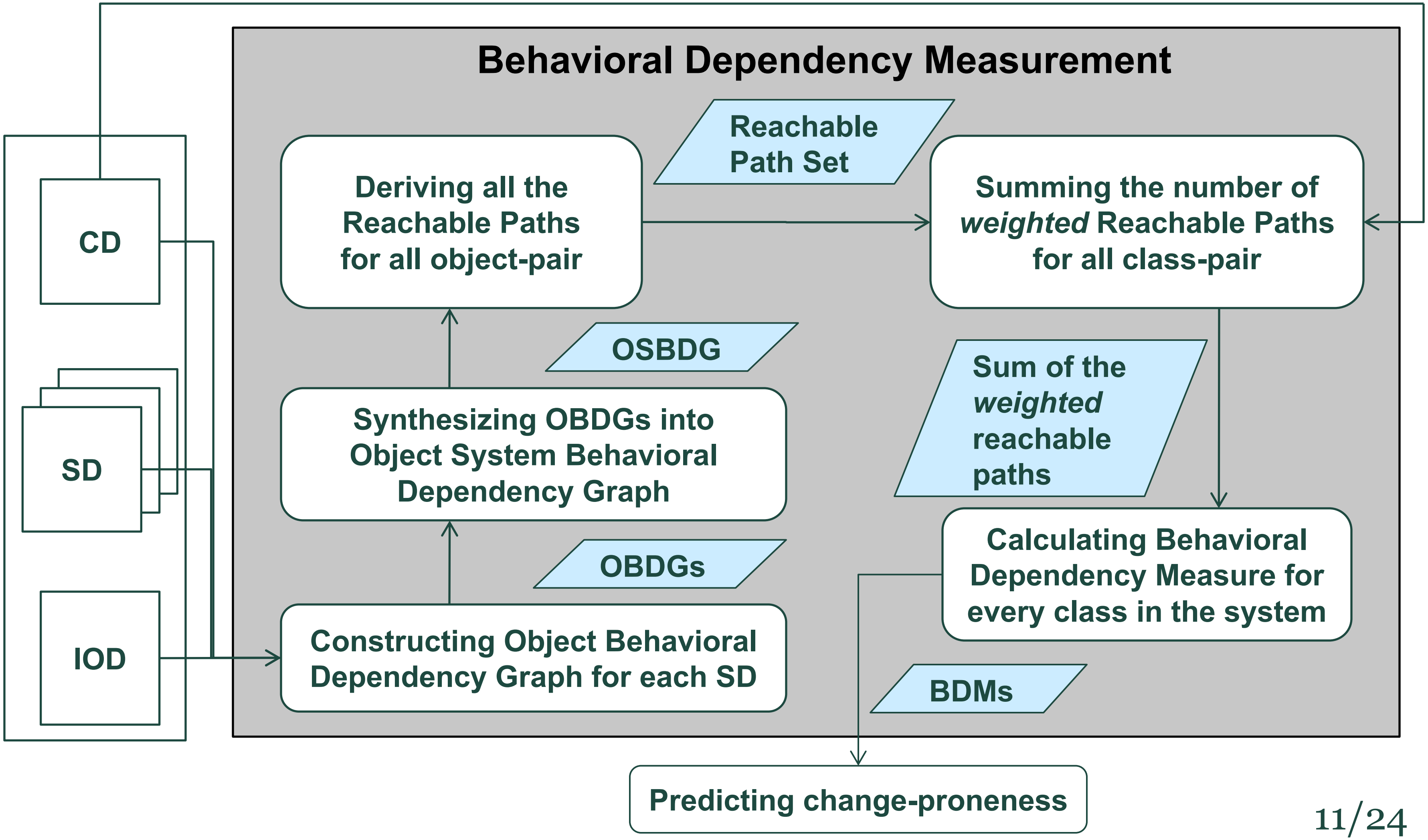
Class Diagram (CD)



Object Behavioral Dependency Graph (OBDG)

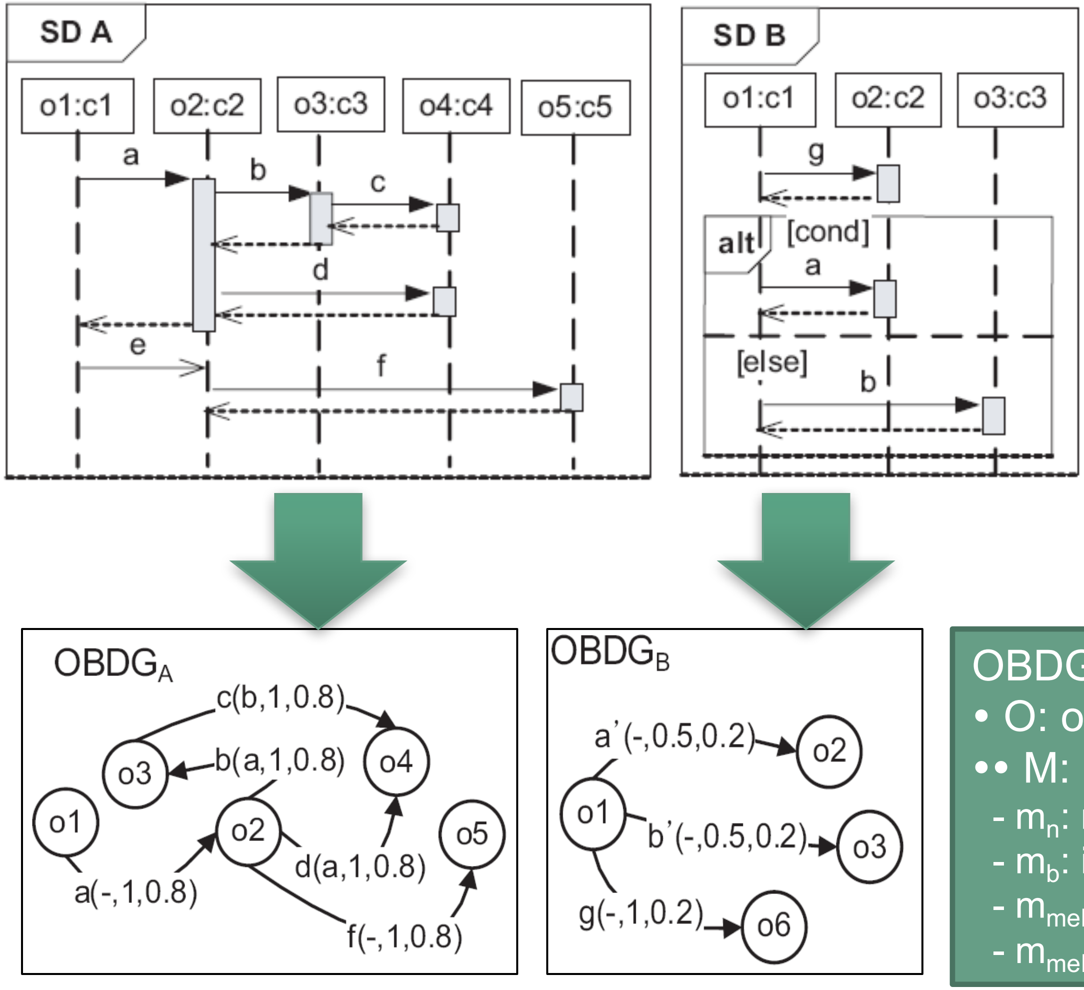
Overview of Our Approach

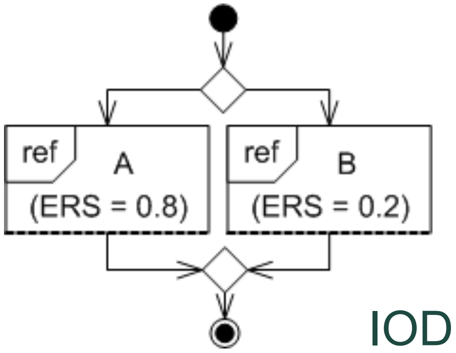
- Model-based change-proneness prediction

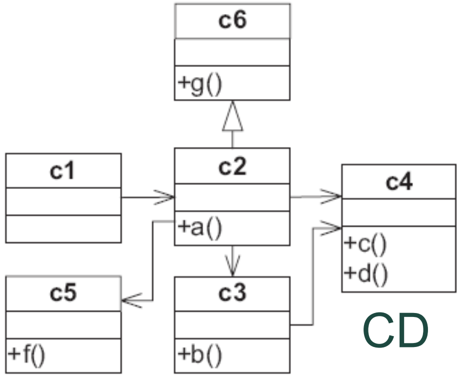


Constructing OBDG and OSBDG (1/2)

❖ Constructing OBDG for each SD



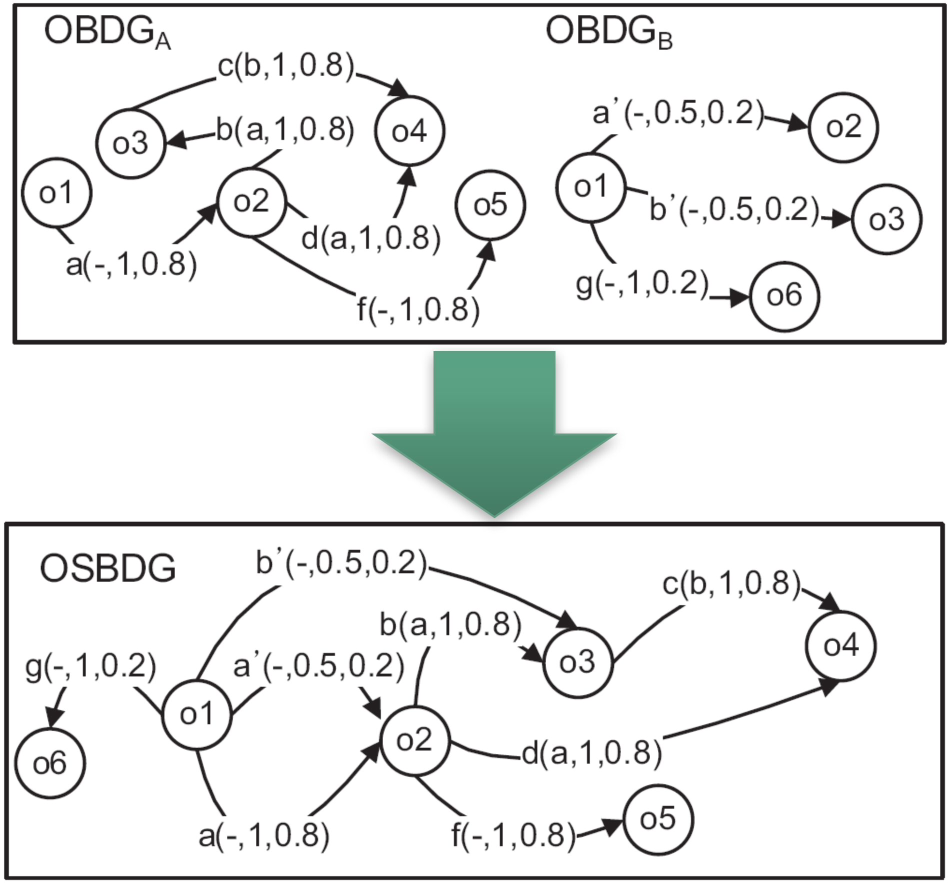




$OBDG_A$ = {O,M}, where
- O: objects in the SD A
- • M: $m_n(m_b,m_{meL},m_{meH})$
  - m_n: message name
  - m_b: instance of a backward navigable message
  - m_{meL}: probabilistic execution rate in SD
  - m_{meH}: expected message execution rate in IOD

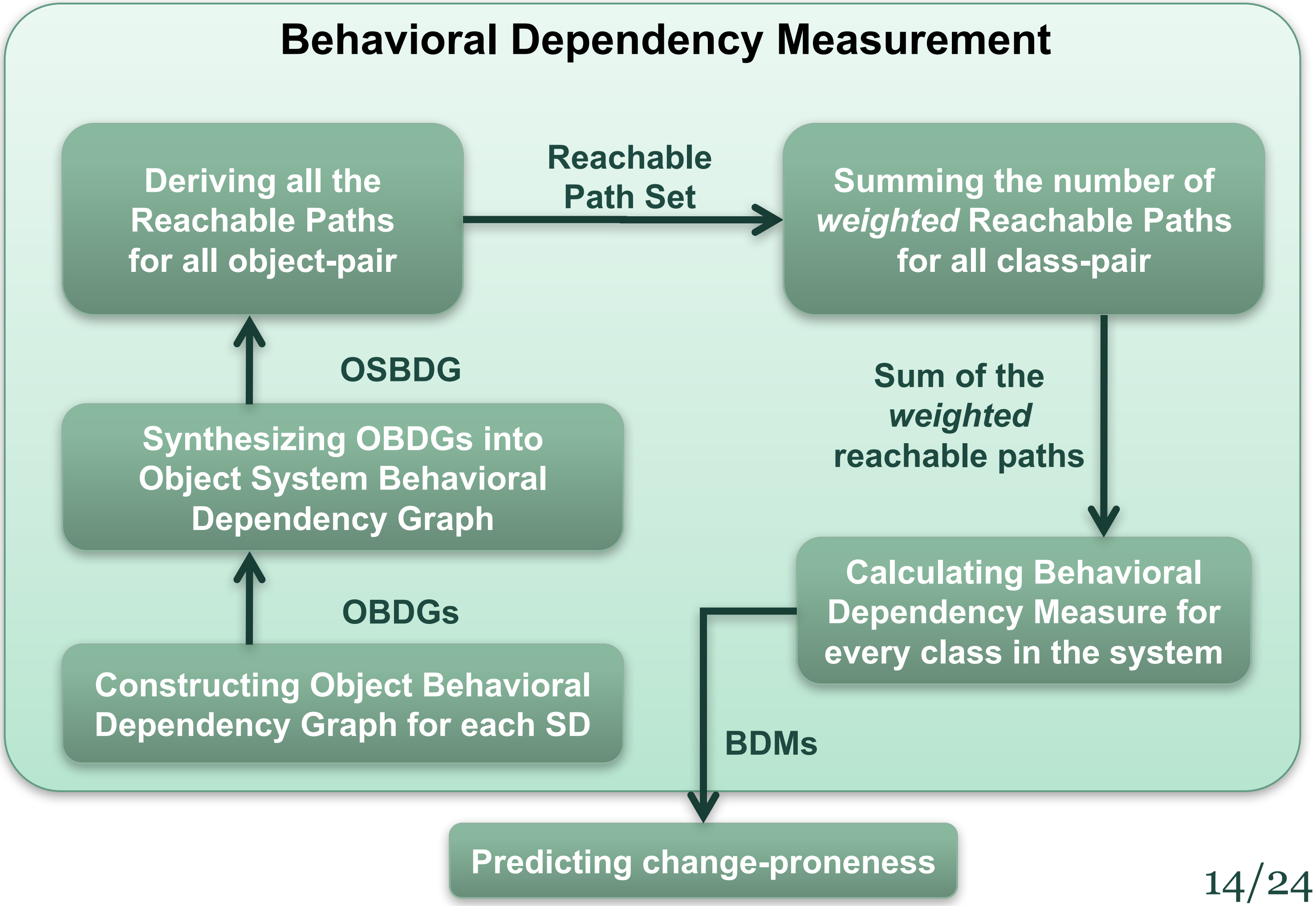
Constructing OBDG and OSBDG (2/2)

- Synthesizing OBDGs into OSBDG



Where we are now

- Model-based change-proneness prediction



Behavioral Dependency Measurement

Constructing Object Behavioral Dependency Graph for each SD

OBDGs

Synthesizing OBDGs into Object System Behavioral Dependency Graph

OSBDG

Deriving all the Reachable Paths for all object-pair

Reachable Path Set

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

Sum of the *weighted* reachable paths

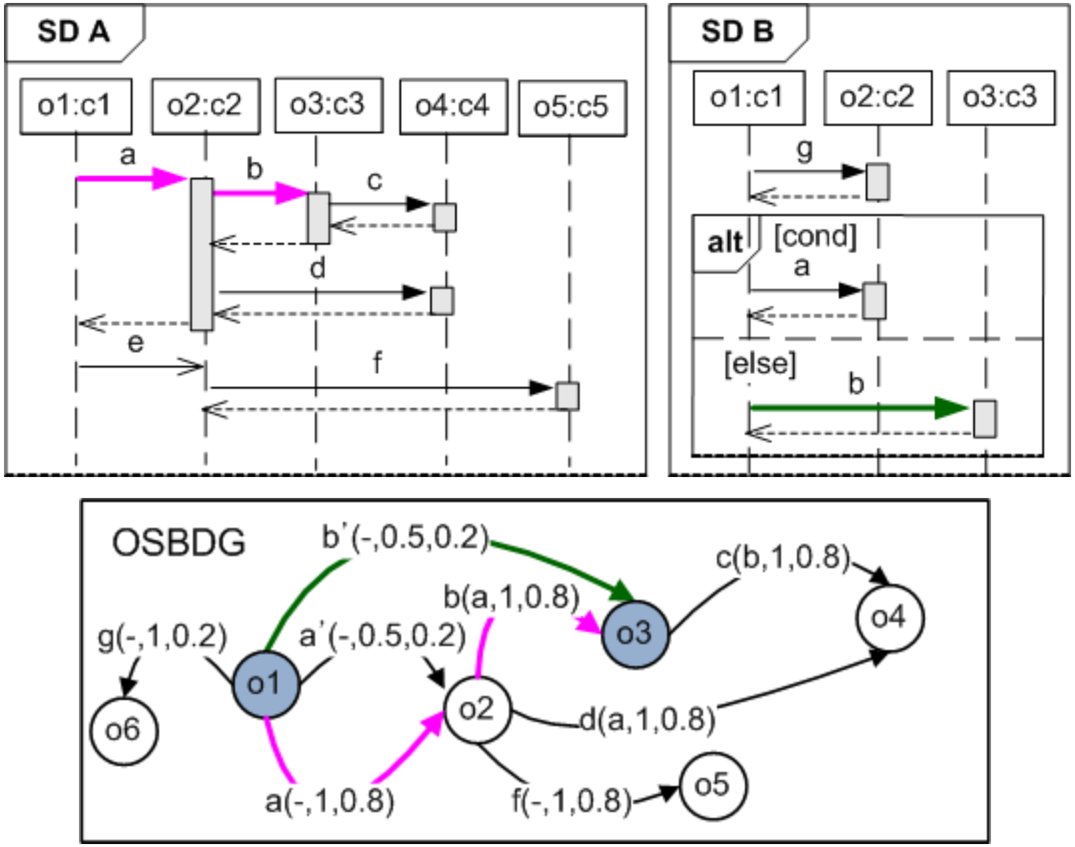
Calculating Behavioral Dependency Measure for every class in the system

BDMs

Predicting change-proneness

Deriving Reachable Paths

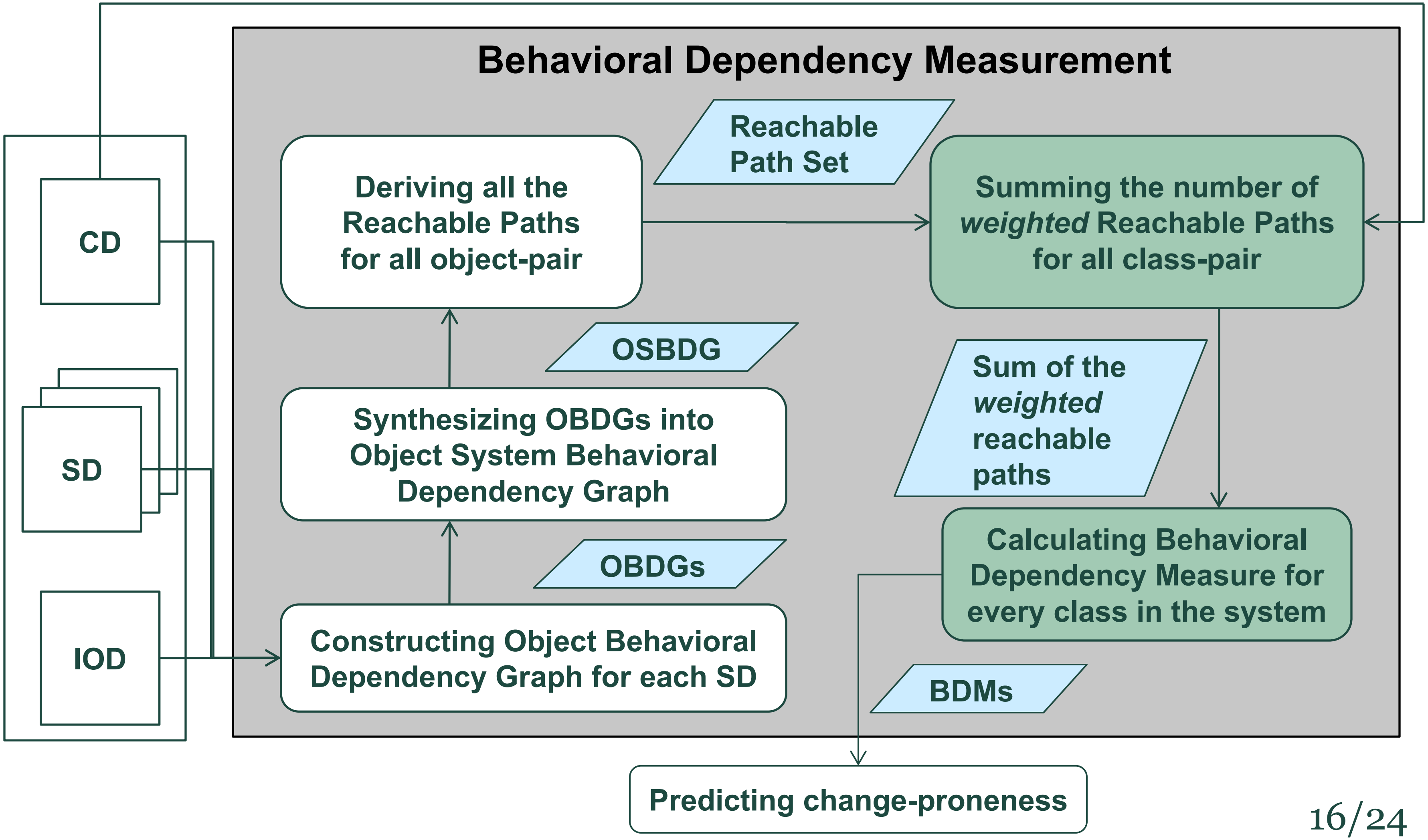
- Deriving all reachable paths for all pair of objects in the system using OSBDG



An example of reachable path set from o1 and o3 : {ab, b'}

Where we are now

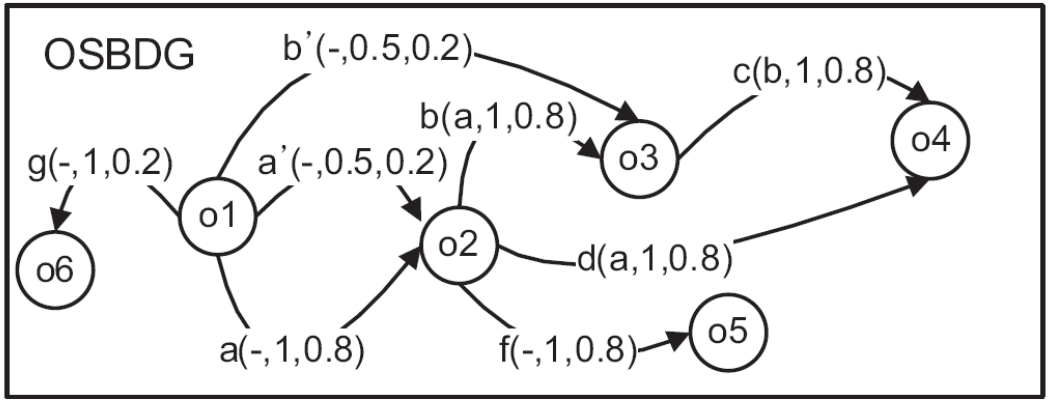
- Model-based change-proneness prediction



Calculating BDM (1/2)

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

$$SumWRP(c1,c2) = \sum_{\forall s \in RPS(o_1,o_2)} DF(s) \times f_{meH} \times f_{meL}$$



Receiver classes

Sender classes

	c1	c2	c3	c4	c5	c6
c1	0	0.9	0.5	0.67	0	0.2
c2	0	0	0.8	1.2	0.8	0
c3	0	0	0	0.4	0	0
c4	0	0	0	0	0	0
c5	0	0	0	0	0	0
c6	0	0	0	0	0	0

Since RPS(o2,o4) = {bc, d}, SumWRP(c1,c2) = (½ x 0.8 x 1) + (1 x 0.8 x 1)

- DF(s) = 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

$$\boxed{BDM(c1)} = \sum_{\forall c_n \in C} SumWRP(c1, c_n)$$

Receiver classes

Sender classes

	c1	c2	c3	c4	c5	c6	BDM(c)
c1	0	0.9	0.5	0.67	0	0.2	2.27
c2	0	0	0.8	1.2	0.8	0	2.8
c3	0	0	0	0.8	0	0	0.8
c4	0	0	0	0	0	0	0
c5	0	0	0	0	0	0	0
c6	0	0	0	0	0	0	0

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

Case Study (1/4)

Goal

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

Experiment Design

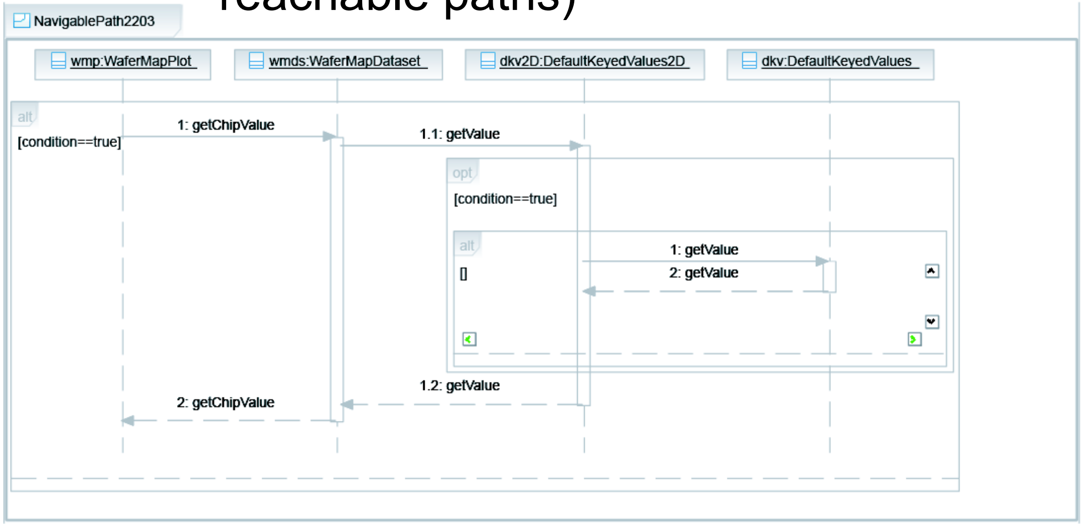
- Make two multivariate regression models with different independent variable set
  - Only C&K metrics* vs. BDM in addition to C&K metrics
- Compare goodness of the fit of those models

* S. Chidamber, C. Kemerer, and C. MIT. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6) : 476–493, 1994.

Case Study (2/4)

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)

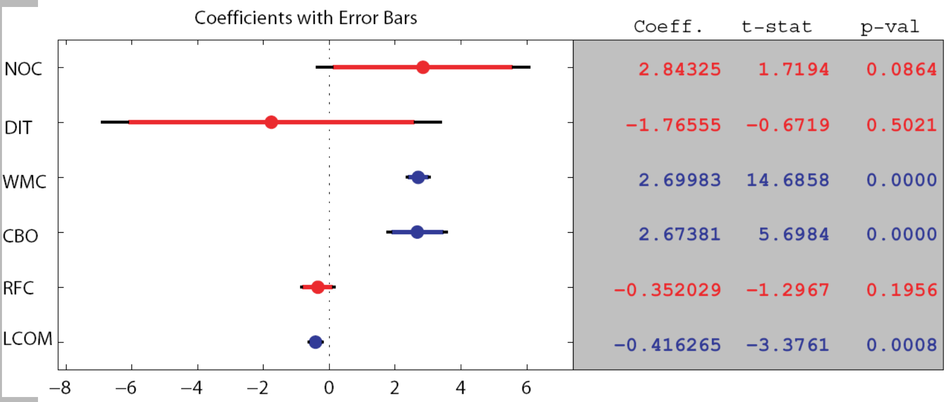
Studied Environment (Cont'd)

- 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

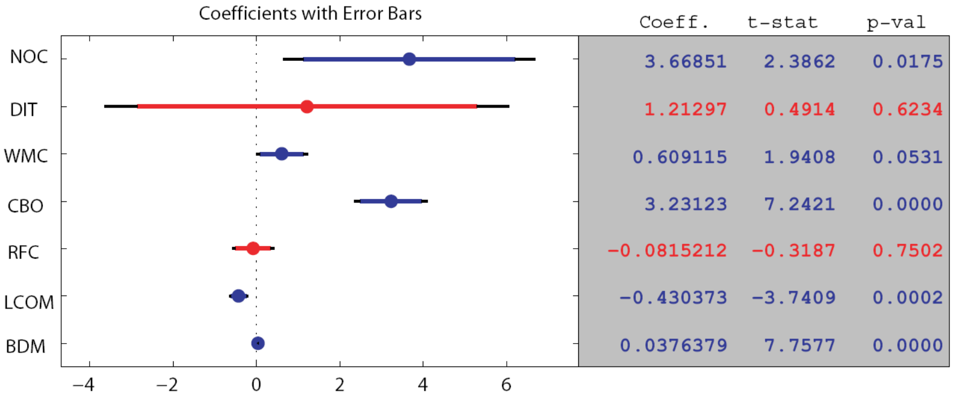
Case Study (4/4)

Results

- Models with only C&K metrics
  - Explains around 56% (adjusted R^2 of <u>0.55</u>)
  - Selected variables
    - WMC (1st), CBO (2nd), and LCOM (3rd)
- Model with BDM in addition to C&K metrics
  - Explains around 64% (adjusted R^2 of <u>0.64</u>)
  - Selected variables
    - WMC (1st), **BDM (2nd)**, CBO (3rd), LCOM (4th), and NOC (5th)



	Coeff.	t-stat	p-val
NOC	2.84325	1.7194	0.0864
DIT	-1.76555	-0.6719	0.5021
WMC	2.69983	14.6858	0.0000
CBO	2.67381	5.6984	0.0000
RFC	-0.352029	-1.2967	0.1956
LCOM	-0.416265	-3.3761	0.0008



	Coeff.	t-stat	p-val
NOC	3.66851	2.3862	0.0175
DIT	1.21297	0.4914	0.6234
WMC	0.609115	1.9408	0.0531
CBO	3.23123	7.2421	0.0000
RFC	-0.0815212	-0.3187	0.7502
LCOM	-0.430373	-3.7409	0.0002
BDM	0.0376379	7.7577	0.0000

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 .

KAIST
SE LAB
KAIST Software Engineering Laboratory