Do Dynamic Object Process Graphs support Program Understanding? – A Controlled Experiment

Jochen Quante

AG Softwaretechnik
FB 3 Mathematik/Informatik
Universität Bremen

18 June 2008
Outline

1. Dynamic Object Process Graphs
2. The Generic Empirical Method
3. Experimental Design
4. Results, Evaluation
Dynamic Object Process Graph (DOPG)

- Projection of Control Flow Graph
- Extracted by dynamic analysis
- Contains only nodes *relevant* for one object
void main () {
    int i = 0;
    Stack s1 = new Stack();
    Stack s2 = Stack.read();
    reverse(s2, s1);
    do {
        s1.pop();
        i = i + 1;
    } while (!s1.empty());
}

void reverse(Stack from, Stack to) {
    while (!from.empty())
        to.push(from.pop());
}
void main () {
    int i = 0;
    Stack s1 = new Stack();
    Stack s2 = Stack.read();
    reverse(s2, s1);
    do {
        s1.pop();
        i = i + 1;
    } while (!s1.empty());
}

void reverse(Stack from,
             Stack to) {
    while (!from.empty())
        to.push(from.pop());
}
void main () {
    int i = 0;
    Stack s1 = new Stack();
    Stack s2 = Stack.read();
    reverse(s2, s1);
    do {
        s1.pop();
        i = i + 1;
    } while (!s1.empty());
}

void reverse(Stack from, 
        Stack to) {
    while (!from.empty())
        to.push(from.pop());
}
void main () {
    int i = 0;
    Stack s1 = new Stack();
    Stack s2 = Stack.read();
    reverse(s2, s1);
    do {
        s1.pop();
        i = i + 1;
    } while (!s1.empty());
}

void reverse(Stack from, Stack to) {
    while (!from.empty())
        to.push(from.pop());
}
Dynamic Object Process Graph – Example

CallJetrisMainFrame.access$23()
CallJetrisMainFrame.moveRight()
CallJetrisMainFrame.access$24()
Figure getMaxRightOffset()
CallJetrisMainFrame.nextMove()
CallJetrisMainFrame.moveDown()
CallJetrisMainFrame.nextMove()
CallJetrisMainFrame.pair()
CallJetrisMainFrame$1.keyPressed()
CallJetrisMainFrame$GridThread.run()
CallJetrisMainFrame.access$4()
CallJetrisMainFrame.nextMove()
Figure setOffset(int, int)
Dynamic Object Process Graph – Example

Presumably useful for program understanding
How to conduct an empirical study: [Prechelt 2006]

1. Decide on ultimate goal
2. Formulate question for the study
3. Characterize the observations sought
4. Design the study
5. Find or create the observation context
6. Observe
7. Analyze observations
8. Interpret results
Decide on ultimate goal:

- Provides context
- Study contributes to this goal
Ultimate Goal

Decide on ultimate goal:

- Provides context
- Study contributes to this goal

Concrete ultimate goal:

Support a maintainer in gaining an understanding of a software system to allow efficient and correct changes.
Research Question

Formulate question for the study:

- Specific goal
- Must be concrete and realistically answerable

Formulate a hypothesis:

- Predicted relationship among the variables being investigated
Formulate question for the study:
- Specific goal
- Must be concrete and realistically answerable

Formulate a hypothesis:
- Predicted relationship among the variables being investigated

Concrete research hypotheses:
- **Time**: The availability of visualized DOPGs reduces the time that is needed to find the answers to program understanding questions when they are in some way related to a given object.
- **Quality**: These questions are also answered less erroneously.
Select an appropriate empirical method:

- Controlled experiment
- Quasi-experiment
- Case study
- Survey
- Benchmark
- Literature study
- ...
Select an appropriate empirical method:

- **Controlled experiment**
- Quasi-experiment
- Case study
- Survey
- Benchmark
- Literature study
- ...
Controlled Experiment: One-Group After-Only Design

Target Population

Selected Sample

Treatment

Measurement

Are the results good?
Controlled Experiment: One-Group After-Only Design

- Target Population
- Selected Sample
- Treatment
- Measurement
- Are the results good?

- Missing any sort of comparison
Controlled Experiment: One-Group Before-After Design

Target Population

Sample

X

Measurement

A

Treatment

X'

Measurement

Is there a difference between X and X'?
Controlled Experiment: One-Group Before-After Design

Does not control for extraneous influences
Controlled Experiment: After-Only Design

- Target Population
- Representative Sample
- Control Group
- Experimental Group
- Independent Variable (Treatment)
- Dependent Variable (Measurement)
- Is there a difference?
Controlled Experiment

- Controlled variation of independent variable(s)
- Measure dependent variable(s)
- Hold extraneous variables constant
Controlled Experiment

• Controlled variation of independent variable(s)
• Measure dependent variable(s)
• Hold extraneous variables constant

⇒ dependent variables measure the effect of the independent variable(s)
⇒ the independent variable must be the only factor that varies systematically
⇒ but does the dependent variable truly reflect the phenomenon under study? (validity)
Extraneous Variables

- History
- Maturation of individuals
- Instrumentation
- Statistical Regression
- Selection
- Mortality
- Participant effect (positive self-presentation)
- Experimenter effect: attributes and expectancies
- Sequencing effect
Techniques for Achieving Constancy

- Randomization
- Matching
- Counterbalancing
- Control of participant effects:
  - Double blind placebo model
  - Deception
  - Independent measurement of the dependent variable
  - Procedural control, e.g. postexperimental inquiry, concurrent probing, think-aloud
- Control of experimenter effects:
  - Blind or partial blind technique
  - Automation
Concrete Study Design

- Between-participants after-only design
  - Two interleaved experiments
  - max. 1 treatment per participant

<table>
<thead>
<tr>
<th></th>
<th>Exp.</th>
<th>DOPG</th>
<th></th>
<th>Exp.</th>
<th>DOPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>A</td>
<td>no</td>
<td>Group 1</td>
<td>B</td>
<td>yes</td>
</tr>
<tr>
<td>Group 2</td>
<td>A</td>
<td>yes</td>
<td>Group 2</td>
<td>B</td>
<td>no</td>
</tr>
<tr>
<td>Group 3</td>
<td>B</td>
<td>no</td>
<td>Group 3</td>
<td>A</td>
<td>yes</td>
</tr>
<tr>
<td>Group 4</td>
<td>B</td>
<td>yes</td>
<td>Group 4</td>
<td>A</td>
<td>no</td>
</tr>
</tbody>
</table>
Concrete Study Design

- Between-participants after-only design
  - Two interleaved experiments
  - max. 1 treatment per participant

<table>
<thead>
<tr>
<th>Group</th>
<th>Exp.</th>
<th>DOPG</th>
<th>Exp.</th>
<th>DOPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>A</td>
<td>no</td>
<td>B</td>
<td>yes</td>
</tr>
<tr>
<td>Group 2</td>
<td>A</td>
<td>yes</td>
<td>B</td>
<td>no</td>
</tr>
<tr>
<td>Group 3</td>
<td>B</td>
<td>no</td>
<td>A</td>
<td>yes</td>
</tr>
<tr>
<td>Group 4</td>
<td>B</td>
<td>yes</td>
<td>A</td>
<td>no</td>
</tr>
</tbody>
</table>

- Required sample size: depends on the effect size; effect size = 0.7, power level 0.8, alpha = 0.05 ⇒ min. 12 participants
Concrete study design

- **Independent variable:**
  - DOPG availability

- **Dependent variables:**
  - Response time to task
  - Correctness of solution
  - Subjective user satisfaction/confidence/productivity

- **Extraneous variables:**
  - User experience: randomize
  - Familiarity with the subject system: don’t allow
  - Instrumentation: automate measurement
  - Selection: don’t restrict, invite all CS students to participate
  - Experimenter effect: use slides and automation
Participants: 27 students from Univ. Bremen
- Respondents to a call
- 1–10 years programming experience (median 5)
- 60% with programming capabilities “above average”
- Largest system: 1–200 KLOC (median 17)
- 50% had worked on others’ code

Environment: 4 identical Linux workstations running Eclipse
- Plugin for displaying/interacting with DOPGs
- Plugin for presenting tasks/questions, capturing answers, and measuring response times

Input materials: Eclipse project, source code, DOPG legend printouts
- Relevant class preselected (known to both groups)
- DOPGs precalculated
Procedure

- Introduction (10 min)
- Training (Talk, 15 min)
  - Program understanding basics: Motivation and approaches
  - Introduction to DOPGs: Construction and example
- Training Tasks (25 min)
  - Goal: get used to environment and nature of tasks
  - Solutions were presented in parallel
- Experiment 1 Tasks (25 min)
- Questionnaire (5 min)
- Experiment 2 Tasks (25 min)
- Questionnaire (5 min)
- Debriefing (10 min)
Experiment Tasks

Experiment 1: ArgoUML [ClassDiagramGraphModel] 320 KLOC

• Which code defines which diagram type is initially opened?
• How is the addition of an element to a diagram done?
• Which class records and keeps the selection history?

Experiment 2: GanttProject [GanttTask] 62 KLOC

• Which code updates the length of a parent task?
• Which component draws the task box?
• Which class keeps dependency information?

No time limit per task; number of tasks unknown
Experiment Tasks

Experiment 1: ArgoUML [ClassDiagramGraphModel] 320 KLOC

- Which code defines which diagram type is initially opened?
- How is the addition of an element to a diagram done?
- Which class records and keeps the selection history?

Experiment 2: GanttProject [GanttTask] 62 KLOC

- Which code updates the length of a parent task?
- Which component draws the task box?
- Which class keeps dependency information?

No time limit per task; number of tasks unknown

These decisions are most critical!
Concrete Threats to Validity

External validity:
- Program representativeness
- Task representativeness
- Experience
- Familiarity with the system
- DOPG experience
- Experimenter effect
- Choice of relevant classes

Internal validity:
- Individual participant differences
- Instrumentation
- Session differences
- Sequencing effects
- Subjects’ perception
Observe

- Actual data collection
- Measure various dependent variables
  - Should be as exact as possible
  - Automation desired
- Interview people or collect filled-in questionnaires
Analyze observations

- Analyze the collected data
- Goal: Answer the study question
- Analyze quantitative data by applied statistics
- Did the observed difference occur by chance?
Statistical analysis

- 99% of all cases
- 95% of all cases
- Frequency of occurrence
- Difference between means
Evaluation: ArgoUML

Response time to Q1 [sec]:

\[ p_t = 0.031, \ p_u = 0.011, \ p_b = 0.035 \]

Share of correct answers:

\[ p_t = 0.001, \ p_u = 0.001, \ p_b = 0.001 \]
Evaluation: GanttProject

Response time to Q1 [sec]:

\[ p_t = 0.813, \quad p_u = 1.000, \quad p_b = 0.812 \]

Share of correct answers:

\[ p_t = 0.511, \quad p_u = 0.599, \quad p_b = 0.530 \]
Evaluation: Rating of Different Tools

ArgoUML

GanttProject
Further Results and Interpretation

Further results:

- DOPGs were considered helpful
- Most participants said they need more time
- No correlation between self-assessment and results
Further Results and Interpretation

Further results:

- DOPGs were considered helpful
- Most participants said they need more time
- No correlation between self-assessment and results

Concerning the hypothesis:

- ArgoUML: null hypothesis rejected ($\alpha = 0.05$)
- GanttProject: null hypothesis not rejected
Further Results and Interpretation

Further results:

- DOPGs were considered helpful
- Most participants said they need more time
- No correlation between self-assessment and results

Concerning the hypothesis:

- ArgoUML: null hypothesis rejected ($\alpha = 0.05$)
- GanttProject: null hypothesis not rejected

Potential reasons:

- Different size/number/structure of DOPGs
  - ArgoUML: 1 DOPG, 167 nodes, 237 edges
  - GanttProject: 3 DOPGs, 66/293/661 nodes, 84/409/963 edges
- Different complexity of tasks
Lessons Learned

- Choice of subject systems and tasks is crucial
- Closeness to reality / session duration vs. number of participants
- Time limits must be carefully chosen
- Single result does not have high credibility
