Supporting Framework Use via Automatically Extracted Concept-Implementation Templates

Abbas Heydarnoori

Krzysztof Czarnecki

Thiago Tonelli Bartolomei

Generative Software Development Lab

University of Waterloo, Canada

Outline

- Introduction and Motivation
- The FUDA Framework Comprehension Technique
- Evaluations
- Concluding Remarks

Introduction and Motivation

Introduction

- *Object-oriented application frameworks* are widely used to develop new applications

- Frameworks provide *domain-specific concepts*
  - *Example: JFace* offers implementation for *context menu* and *tree viewer*

- Framework-based applications are constructed by writing completion code that instantiates these concepts

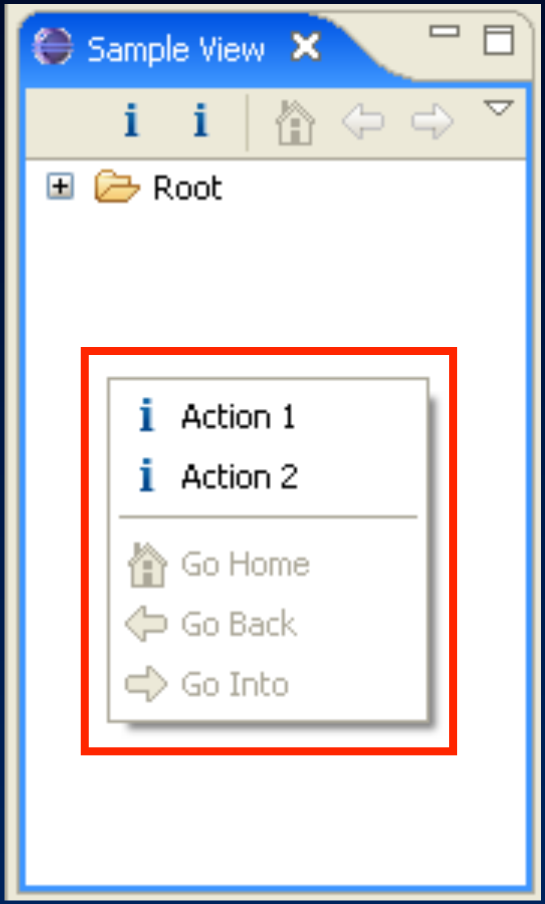
Main Difficulties of Frameworks

- Complex and difficult to learn APIs
- Lack of manuals and documentation

Proposed Solution

- Apply the *Monkey See/Monkey Do* Rule *[Gamma et al., 2004]*
  - "Use existing framework applications as a guide to develop new applications"
- Code difficult to find due to scattering and tangling

Motivating Example



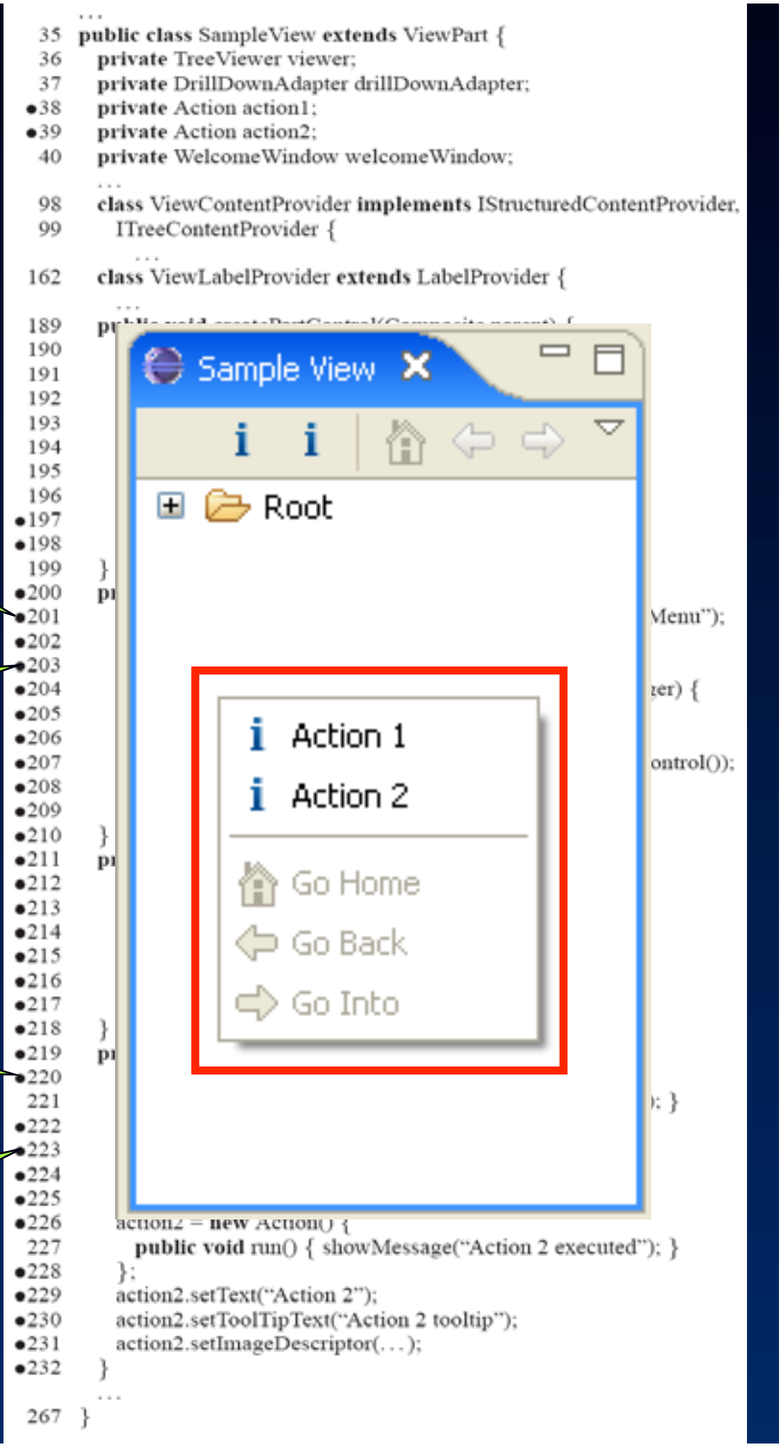
Motivating Example

Instantiating Framework Classes

Implementing Interfaces

Sub-classing Framework Classes

Calling Framework Methods



```
...
public class SampleView extends ViewPart {
  private TreeViewer viewer;
  private DrillDownAdapter drillDownAdapter;
  private Action action1;
  private Action action2;
  private WelcomeWindow welcomeWindow;
  ...
  class ViewContentProvider implements IStructuredContentProvider,
    ITreeContentProvider {
    ...
  class ViewLabelProvider extends LabelProvider {
    ...
  pu
  }
  pr
                                                         Menu");
                                                         er) {
                                                         ontrol());
  }
  pr
  }
  pr
                                                         ); }
    action2 = new Action() {
      public void run() { showMessage("Action 2 executed"); }
    };
    action2.setText("Action 2");
    action2.setToolTipText("Action 2 tooltip");
    action2.setImageDescriptor(...);
  }
  ...
}
```

Related Work

- *Framework usage comprehension tools*, such as *Strathcona* *[ICSE'05]* and *FrUiT* *[ETX'06]*
  - Apply static analyses
  - Aim fine-grained API elements

- *Concept location tools*, such as *SNIAFL* *[ICSE'04]* and *SITIR* *[ASE'07]*
  - Unaware of a framework API
  - Results contain application-specific instructions

FUDA Framework Comprehension Technique

FUDA Framework Comprehension Technique

- Automatically extracts *implementation templates* for framework-provided concepts
  - *Concept Implementation Template:* A Java-like representation of the implementation steps that are necessary to instantiate a given concept

A Sample Template

Basic Steps

Packages to Import

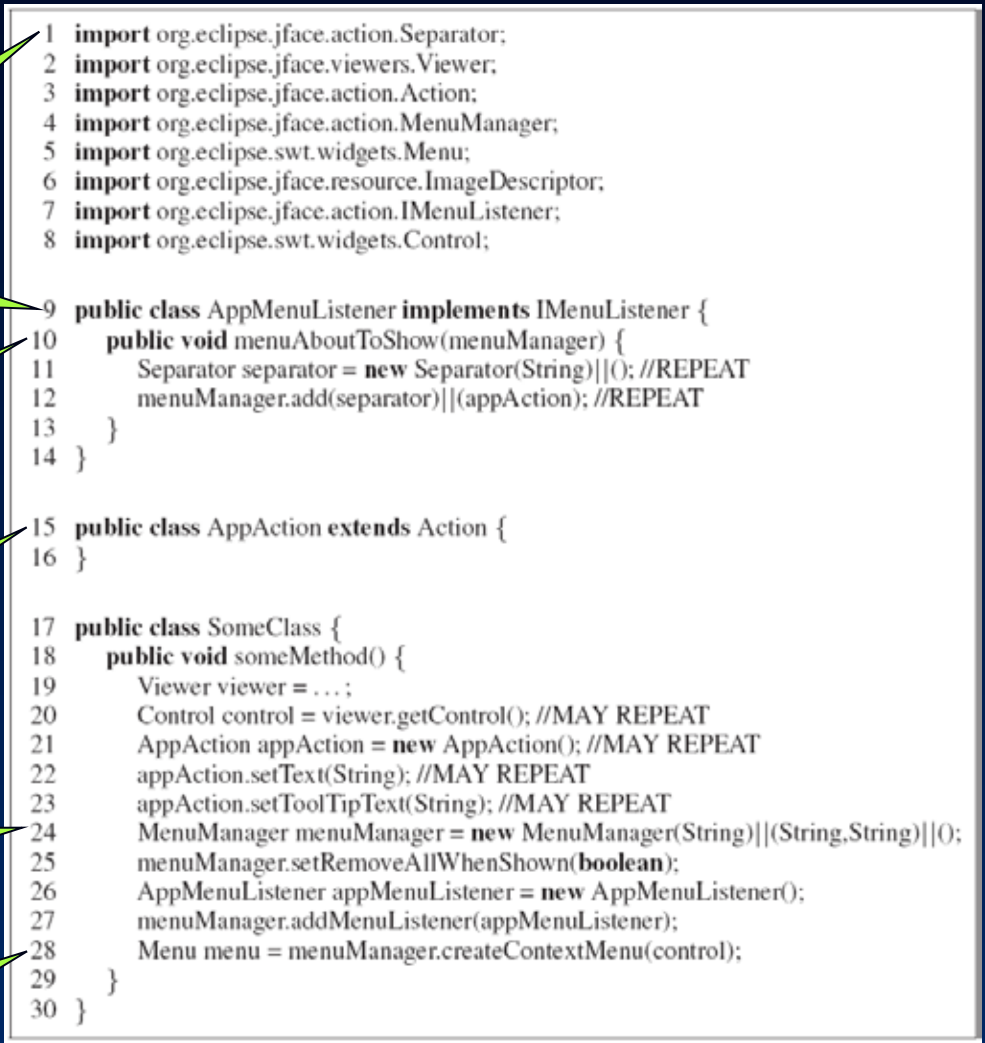
Interfaces to Implement

Methods to Implement

Classes to Subclass

Objects to Create

Methods to Call



```
import org.eclipse.jface.action.Separator;
import org.eclipse.jface.viewers.Viewer;
import org.eclipse.jface.action.Action;
import org.eclipse.jface.action.MenuManager;
import org.eclipse.swt.widgets.Menu;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.jface.action.IMenuListener;
import org.eclipse.swt.widgets.Control;

public class AppMenuListener implements IMenuListener {
    public void menuAboutToShow(menuManager) {
        Separator separator = new Separator(String)||(); //REPEAT
        menuManager.add(separator)||(appAction); //REPEAT
    }
}

public class AppAction extends Action {
}

public class SomeClass {
    public void someMethod() {
        Viewer viewer = . . . ;
        Control control = viewer.getControl(); //MAY REPEAT
        AppAction appAction = new AppAction(); //MAY REPEAT
        appAction.setText(String); //MAY REPEAT
        appAction.setToolTipText(String); //MAY REPEAT
        MenuManager menuManager = new MenuManager(String)||(String,String)||();
        menuManager.setRemoveAllWhenShown(boolean);
        AppMenuListener appMenuListener = new AppMenuListener();
        menuManager.addMenuListener(appMenuListener);
        Menu menu = menuManager.createContextMenu(control);
    }
}
```

Additional Information

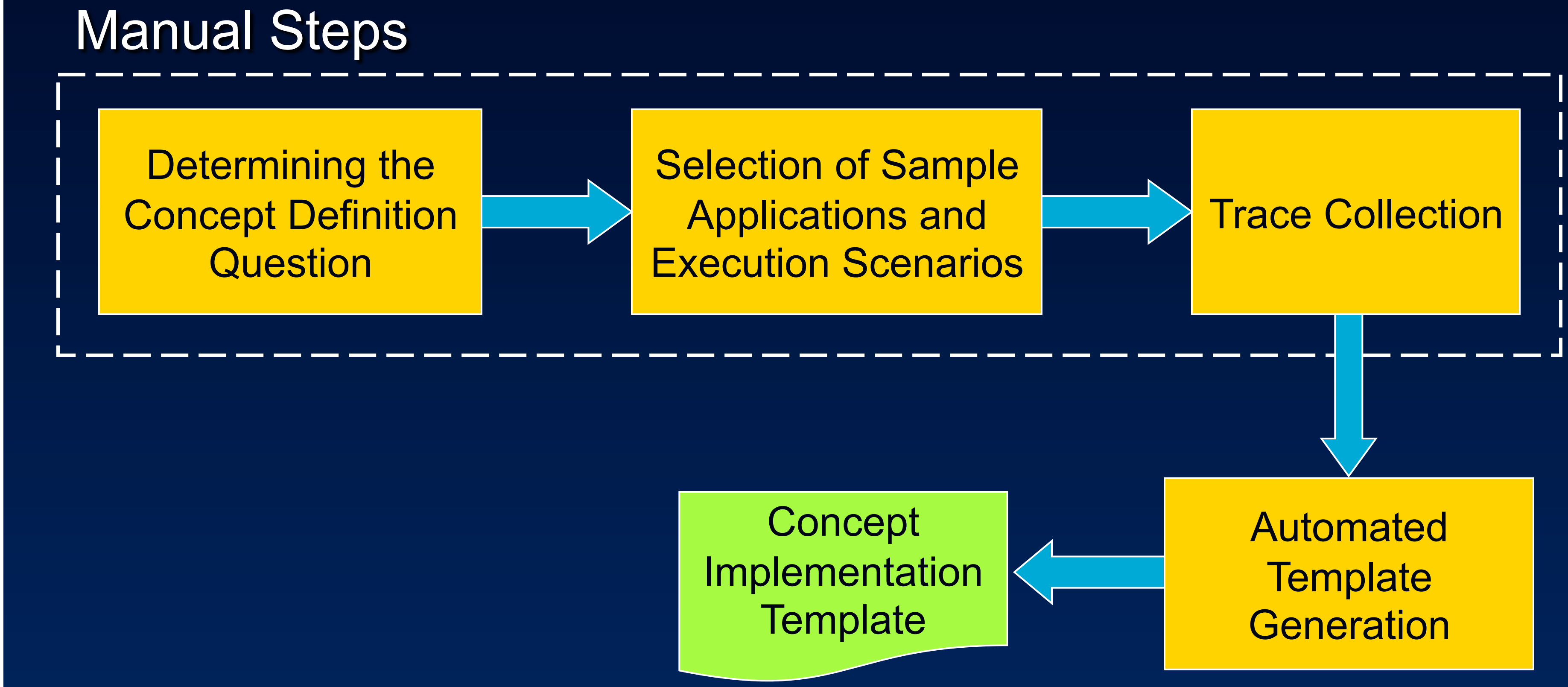
Call Nesting

Order of Calls

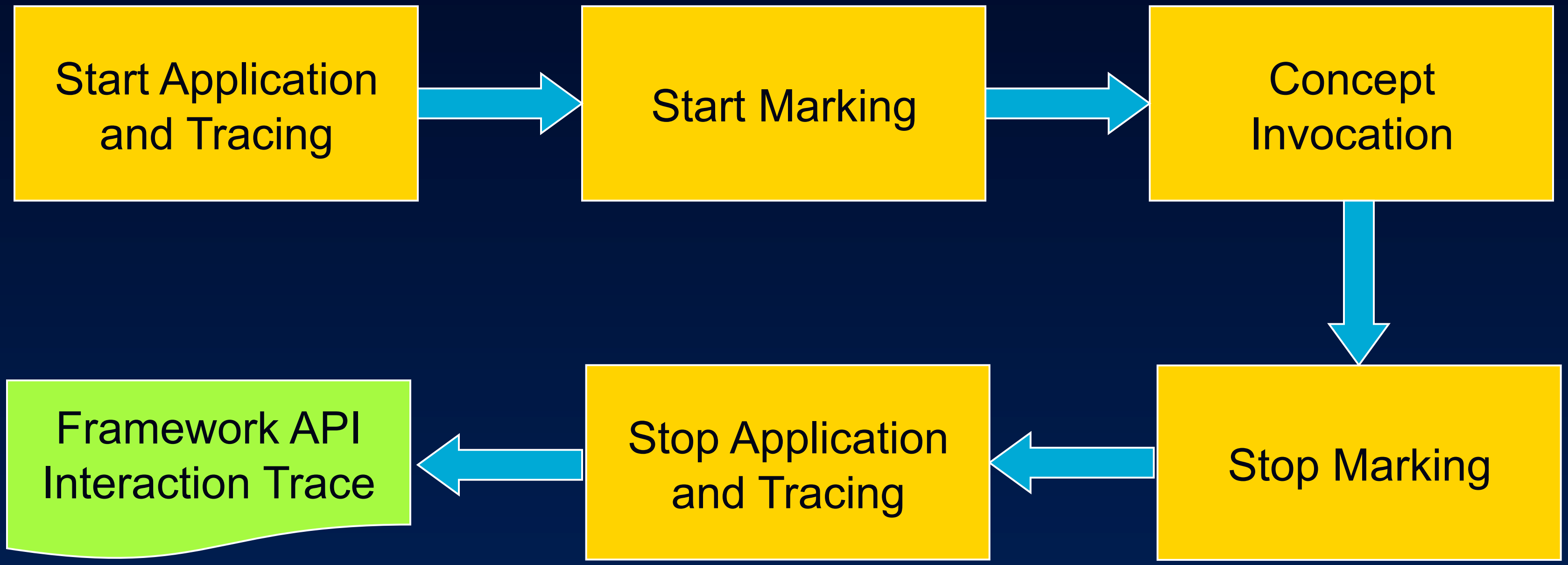
Object Passing Patterns

Statement Repetition Info.

The FUDA Approach Overview

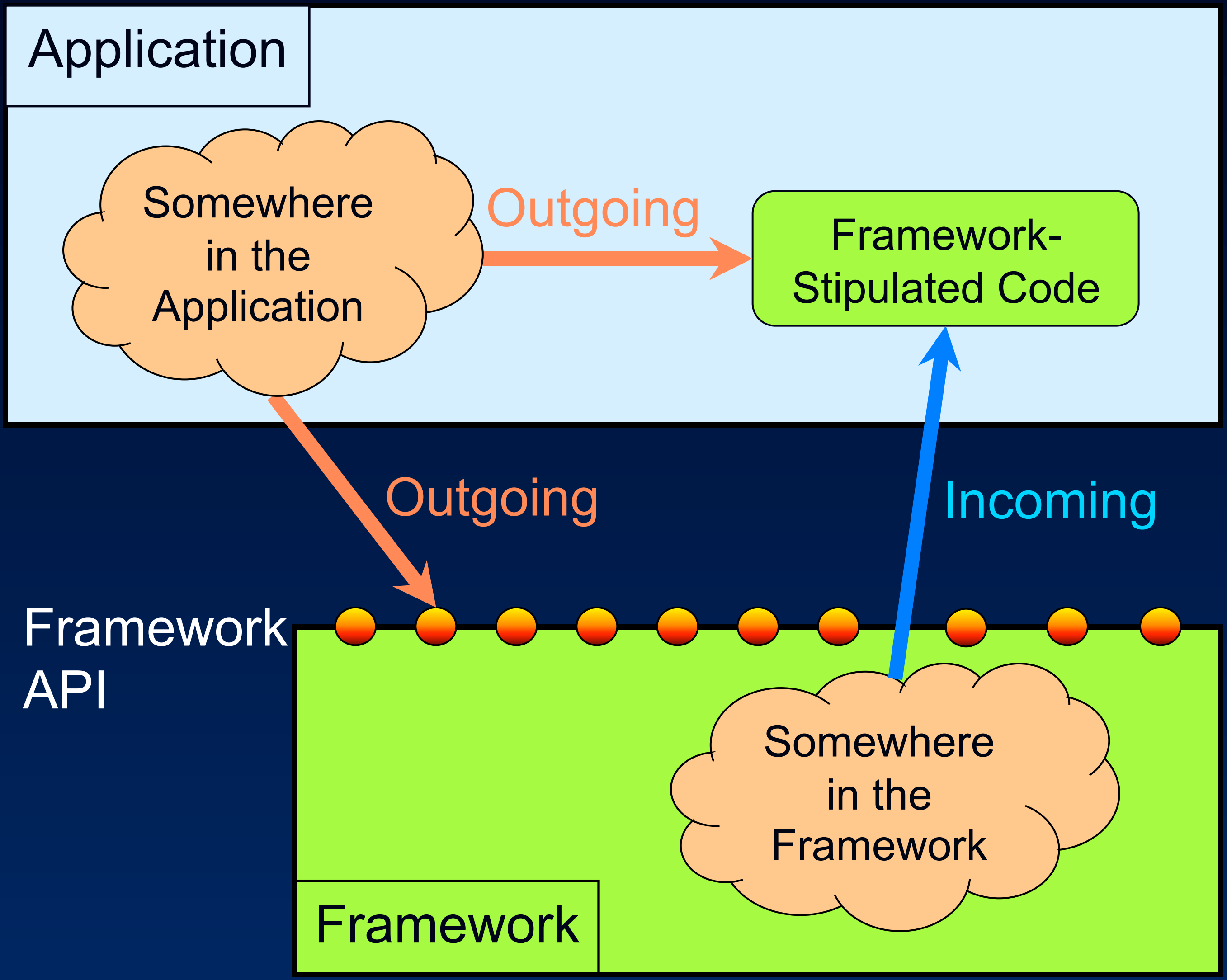

Manual Steps
Determining the Concept Definition Question
Selection of Sample Applications and Execution Scenarios
Trace Collection
Automated Template Generation
Concept Implementation Template

Trace Collection


Start Application and Tracing
Start Marking
Concept Invocation
Stop Marking
Stop Application and Tracing
Framework API Interaction Trace

- Traces only the calls at the framework boundary
  - The API trace consists of *API interaction events*

Direction of Events



Application

Somewhere in the Application

Outgoing

Framework-Stipulated Code

Outgoing

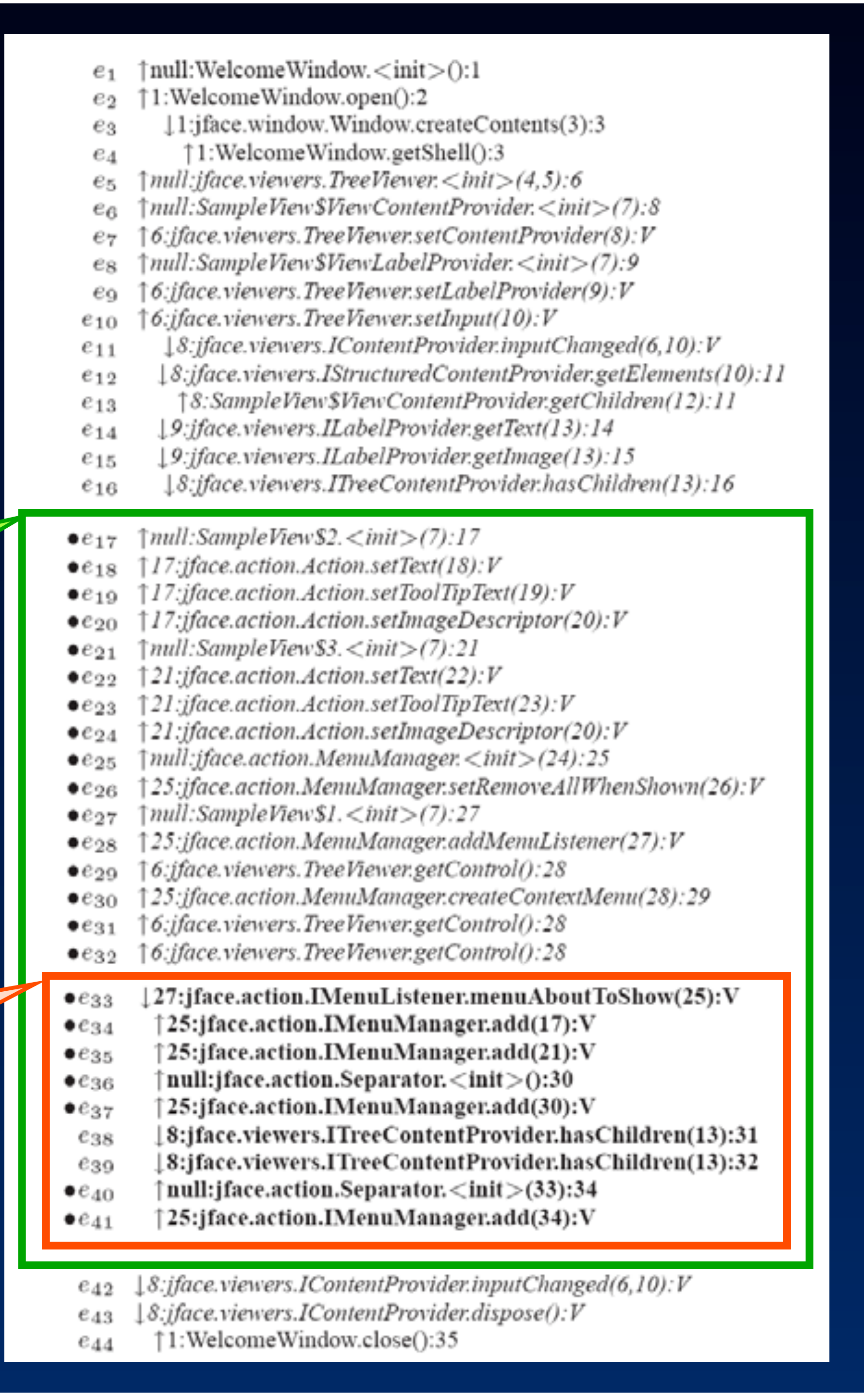
Incoming

Framework API

Somewhere in the Framework

Framework

A Sample API Trace for the Concept Context Menu



```
 e1   ↑null:WelcomeWindow.<init>():1
 e2   ↑1:WelcomeWindow.open():2
        ↓1:jface.window.Window.createContents(3):3
 e4       ↑1:WelcomeWindow.getShell():3
 e5   ↑null:jface.viewers.TreeViewer.<init>(4,5):6
 e6   ↑null:SampleView$ViewContentProvider.<init>(7):8
 e7   ↑6:jface.viewers.TreeViewer.setContentProvider(8):V
 e8   ↑null:SampleView$ViewLabelProvider.<init>(7):9
 e9   ↑6:jface.viewers.TreeViewer.setLabelProvider(9):V
 e10  ↑6:jface.viewers.TreeViewer.setInput(10):V
 e11    ↓8:jface.viewers.IContentProvider.inputChanged(6,10):V
 e12   ↓8:jface.viewers.IStructuredContentProvider.getElements(10):11
 e13      ↑8:SampleView$ViewContentProvider.getChildren(12):11
 e14   ↓9:jface.viewers.ILabelProvider.getText(13):14
 e15   ↓9:jface.viewers.ILabelProvider.getImage(13):15
 e16    ↓8:jface.viewers.ITreeContentProvider.hasChildren(13):16
•e17  ↑null:SampleView$2.<init>(7):17
•e18  ↑17:jface.action.Action.setText(18):V
•e19  ↑17:jface.action.Action.setToolTipText(19):V
•e20  ↑17:jface.action.Action.setImageDescriptor(20):V
•e21  ↑null:SampleView$3.<init>(7):21
•e22  ↑21:jface.action.Action.setText(22):V
•e23  ↑21:jface.action.Action.setToolTipText(23):V
•e24  ↑21:jface.action.Action.setImageDescriptor(20):V
•e25  ↑null:jface.action.MenuManager.<init>(24):25
•e26  ↑25:jface.action.MenuManager.setRemoveAllWhenShown(26):V
•e27  ↑null:SampleView$1.<init>(7):27
•e28  ↑25:jface.action.MenuManager.addMenuListener(27):V
•e29  ↑6:jface.viewers.TreeViewer.getControl():28
•e30  ↑25:jface.action.MenuManager.createContextMenu(28):29
•e31  ↑6:jface.viewers.TreeViewer.getControl():28
•e32  ↑6:jface.viewers.TreeViewer.getControl():28
•e33  ↓27:jface.action.IMenuListener.menuAboutToShow(25):V
•e34    ↑25:jface.action.IMenuManager.add(17):V
•e35    ↑25:jface.action.IMenuManager.add(21):V
•e36    ↑null:jface.action.Separator.<init>():30
•e37    ↑25:jface.action.IMenuManager.add(30):V
 e38    ↓8:jface.viewers.ITreeContentProvider.hasChildren(13):31
 e39    ↓8:jface.viewers.ITreeContentProvider.hasChildren(13):32
•e40    ↑null:jface.action.Separator.<init>(33):34
•e41    ↑25:jface.action.IMenuManager.add(34):V
 e42  ↓8:jface.viewers.IContentProvider.inputChanged(6,10):V
 e43  ↓8:jface.viewers.IContentProvider.dispose():V
 e44    ↑1:WelcomeWindow.close():35
```

Events involved in the implementation of the context menu

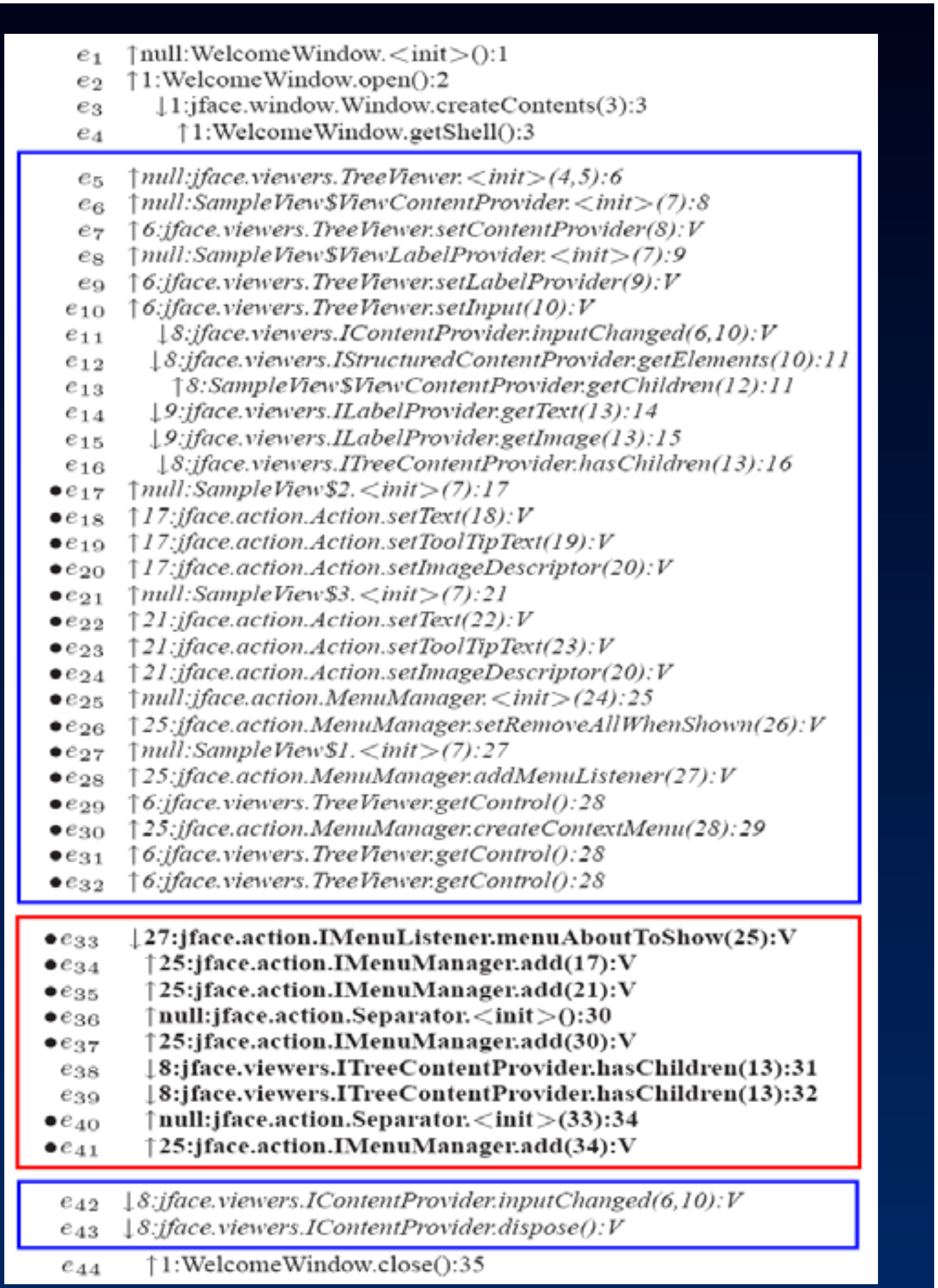
The marked events when the context menu is invoked

API Trace Slicing

- Identifies relevant events before and after the marked region
  - Related events use common objects as targets, parameters, or return values

Sample Sliced API Trace

- An approximation of the actual dependencies among API calls
  - Could have both false positives and false negatives



```
 e1   ↑null:WelcomeWindow.<init>():1
 e2   ↑1:WelcomeWindow.open():2
 e3     ↓1:jface.window.Window.createContents(3):3
 e4       ↑1:WelcomeWindow.getShell():3

 e5   ↑null:jface.viewers.TreeViewer.<init>(4,5):6
 e6   ↑null:SampleView$ViewContentProvider.<init>(7):8
 e7   ↑6:jface.viewers.TreeViewer.setContentProvider(8):V
 e8   ↑null:SampleView$ViewLabelProvider.<init>(7):9
 e9   ↑6:jface.viewers.TreeViewer.setLabelProvider(9):V
 e10  ↑6:jface.viewers.TreeViewer.setInput(10):V
 e11    ↓8:jface.viewers.IContentProvider.inputChanged(6,10):V
 e12    ↓8:jface.viewers.IStructuredContentProvider.getElements(10):11
 e13      ↑8:SampleView$ViewContentProvider.getChildren(12):11
 e14    ↓9:jface.viewers.ILabelProvider.getText(13):14
 e15    ↓9:jface.viewers.ILabelProvider.getImage(13):15
 e16    ↓8:jface.viewers.ITreeContentProvider.hasChildren(13):16
•e17  ↑null:SampleView$2.<init>(7):17
•e18  ↑17:jface.action.Action.setText(18):V
•e19  ↑17:jface.action.Action.setToolTipText(19):V
•e20  ↑17:jface.action.Action.setImageDescriptor(20):V
•e21  ↑null:SampleView$3.<init>(7):21
•e22  ↑21:jface.action.Action.setText(22):V
•e23  ↑21:jface.action.Action.setToolTipText(23):V
•e24  ↑21:jface.action.Action.setImageDescriptor(20):V
•e25  ↑null:jface.action.MenuManager.<init>(24):25
•e26  ↑25:jface.action.MenuManager.setRemoveAllWhenShown(26):V
•e27  ↑null:SampleView$1.<init>(7):27
•e28  ↑25:jface.action.MenuManager.addMenuListener(27):V
•e29  ↑6:jface.viewers.TreeViewer.getControl():28
•e30  ↑25:jface.action.MenuManager.createContextMenu(28):29
•e31  ↑6:jface.viewers.TreeViewer.getControl():28
•e32  ↑6:jface.viewers.TreeViewer.getControl():28

•e33  ↓27:jface.action.IMenuListener.menuAboutToShow(25):V
•e34    ↑25:jface.action.IMenuManager.add(17):V
•e35    ↑25:jface.action.IMenuManager.add(21):V
•e36    ↑null:jface.action.Separator.<init>():30
•e37    ↑25:jface.action.IMenuManager.add(30):V
 e38    ↓8:jface.viewers.ITreeContentProvider.hasChildren(13):31
 e39    ↓8:jface.viewers.ITreeContentProvider.hasChildren(13):32
•e40    ↑null:jface.action.Separator.<init>(33):34
•e41    ↑25:jface.action.IMenuManager.add(34):V

 e42  ↓8:jface.viewers.IContentProvider.inputChanged(6,10):V
 e43  ↓8:jface.viewers.IContentProvider.dispose():V
 e44    ↑1:WelcomeWindow.close():35
```

Event Generalization

- Allows comparing traces in terms of framework API types

- Replaces application-specific names with appropriate framework names

- A static analysis on the type hierarchy of the event's target

Fact Extraction and Template Generation

- Extracting facts about the call occurrences, nesting, and dependency
- Determining *common facts* across traces
- Template generation from common facts

Evaluations

Template Extraction Evaluation

Template Usage Evaluation

Template Extraction Evaluation

- *Evaluation Hypothesis:* FUDA can extract templates with high precision and recall from only two traces and two sample applications
  - Aimed to keep the number of traces minimal

Selection of Frameworks and Concepts

- Four complex, widely-used frameworks:
  - Eclipse, JFace, GEF, Java2D
- Fourteen concepts:
  - Six based on prior knowledge
  - Eight from developer forums
  - Covered different characteristics: *scope*, *slicing*, *frequency*, *complexity*, and *atomicity*

Selection of Sample Applications

- Two applications per concept from different sources
  - Available at hand
  - Packaged with the desired framework
  - Online repositories
  - Suggested by others

Experiment Performance

- Prototyped FUDA
  - *FUDA Profiler*
    - Provides a GUI for collecting marked traces
    - Uses *AspectJ* to instrument applications
  - *FUDA Analyzer*
    - An Eclipse plug-in for generating templates out of traces

Calculation of Precision and Recall

- Requires a reference
  - Created based on prior experience, framework documentation, and/or actual implementation

Results

- *Precision:* 59% - 100%
    - Presented instructions being correct
- *Recall:* 79% - 100%
    - Required instructions being presented
- Application similarities caused false positives
- Slicing eliminated 18% - 80% of false positives
    - Mainly useful for similar applications
    - No effect for different applications

Template Usage Evaluation

- Evaluated the usage of templates by developers in the implementation of concepts
  - Asked developers to use either *documentation* or *templates*

Research Questions

- ***Q_1:*** Are templates as effective as documentation in aiding the developers?
  - If yes, they can serve as a substitute when no documentation is available
- ***Q_2:*** What is the influence of template quality and its usage strategies on the quality of resulting implementations?

Subjects

- Recruited twelve subjects
  - A mixture of students and professionals
  - Highly skilled Java programmers
  - Except one, all had industry experience

Task Assignment

- Assigned two tasks to each subject:
  - One simple, one complex
  - One using template, one using documentation
- Random and balanced over the concept complexity and documentation aid
- Constrained by prior knowledge of the concepts

Data Analysis

- Quantitatively analyzed via:
  - Statistical analyses of development times

- Qualitatively analyzed via:
  - Inspection and execution of resulting implementations
  - Careful examination of questionnaires and interviews

Quantitative Analysis Results

- The choice of documentation aid had little influence on the development time
  - Statistical analysis failed in providing evidence that templates and documentation are different (or equivalent) in providing aid
  - However:
    - The observed differences due to documentation aid were small
    - The concept complexity had much greater impact on productivity

Qualitative Analysis Results

- Only two buggy implementations
  - One for each documentation aid

- All except one used templates together with sample applications
  - That subject had a buggy implementation
  - Use templates with sample applications

Concluding Remarks

Strengths and Weaknesses of the Approach

Conclusions

Strengths of the Approach

- Templates and documentation were similarly effective in the experiment
  - Not statistically significant; larger experiment needed
- Highly automated
- Needs only a few sample applications
- Traces only the API interactions
- Dynamically detects the actual API elements involved

Weaknesses of the Approach

- Results depend on sample applications
- Designing concept invoking scenarios not always obvious
- Setting up the runtime environment could be challenging

Thank You!

Questions?

Contact Information:

Generative Software Development Lab
University of Waterloo, Canada

gsd@gsd.uwaterloo.ca