Recovery and Migration of Application Logic from Legacy Systems

Wiktor Nowakowski, Michał Śmiałek, Albert Ambroziewicz, Norbert Jarzębowski, Tomasz Straszak

Warsaw University of Technology

Krajowa Konferencja Inżynierii Oprogramowania
Kraków, Poland, September 10-13, 2012
Overview

- Idea & Solution overview
- Essence
- Recovery from legacy system
- Migration to modern technology system
- Summary
Idea

- Software recycling
- Developed within the REMICS project
- Reuse & Migration of Legacy Software Systems
- 7th Framework Program, Information & Communication Technologies
Solution overview

GUI-ripping tool

TALE+ReDSeeDS

Code

VSomeFormClass

public VCourseForm() {
    /* ... */
    jbnClickHere = new JButton("Click Here");
    this.add(jbnClickHere);
    jbnOK.addActionListener(
        /* ... */
        jbnClickHereActionPerformed(evt);
        /* ... */
    );
}

private void jbnClickHereActionPerformed(java.awt.event.ActionEvent evt) {
    cAddNewCourse.clicksClickHereButton();
    
}
Recovery
Migration

Specification
- Use Case
- Use Case
- Use Case
- Use Case
- Use Case

Transformation

Code
- Class
- Class
- Class
- Class
Metodology: Process

Legacy system → GUI-ripping Tool → XML scripts

Tooling framework

Initial RSL-AL model → TALE Recovery Engine → ReDSeeDS RSL-AL Editor → MOLA Transformation Engine

Refined RSL-AL model → ReDSeeDS RSL-AL Editor

Target system structure models → MOLA Transformation Engine

XML scripts

RSL-AL metamodel

UML/SoaML metamodel
Essence
"The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions. [...] such a conceptual construct is the same under many different representations."*

"[...] accidental tasks arise in representing the construct in language."*

* F.P. Brooks, "No silver bullet: Essence and accidents of software engineering"
The „essence” in legacy systems

Some characteristics of legacy systems:
- poorly structured (e.g. complex monolithic structure)
- technology obsolescence (poor interoperability)
- loss of knowledge (both technical and business)

How to capture the essence of a legacy systems?
How to cater this essence for new functionality?
How to migrate it into a new technology?
Recovery from legacy system
Solution overview

GUI-ripping tool

TALE+ReDSeeDS

Code

public VCourseForm() {
    /* ... */
    jbnClickHere = new JButton("Click Here");
    this.add(jbnClickHere);
    jbnOK.addActionListener(
    /* ... */
        jbnClickHereActionPerformed(evt);
    /* ... */
    );
}

private void jbnClickHereActionPerformed(java.awt.event.ActionEvent evt) {
    cAddNewCourse.clicksClickHereButton();
    }

User clicks "Click here" button
GUI-ripping tool

- IBM Rational Functional Tester (RFT)
- Tool for automated testing of software applications
- Supports a broad range of applications such as Web-based, .Net, Java, terminal emulator based applications, Siebel, SAP, PowerBuilder, AJAX, Adobe Flex, Dojo Toolkit, GEF, Adobe PDF documents
- Recording mechanism creates a test script from the actions
- Object map contains information about GUI elements
From test scripts to requirements

1. User selects New entry from BibTex
2. System shows Select entry type
3. User selects Book
4. System shows JabRef - untitled
5. User enters JabRef - untitled
TALE & ReDSeeDS

- Eclipse based (perspectives)
- Integrated use cases, notions and diagrams editors
- Implements RSL metamodel
- Notions with meanings connected to WordNet database
- Enables requirements to code transformations
- Enables model export to Enterprise Architect
Recovered Notions

**Requirements**

Add new book
Precondition: Database is opened
1. User *selects* New entry option
2. System *shows* Select entry type dialog
3. User *selects* Book option
4. System *shows* Book entry form
5. User *fills* Book entry form with Book data
6. System *saves* Book
3. **invoke**: Delete entry

Delete entry

**Requirement representations**

Start

1. User selects New entry option
2. System shows Select entry type dialog
3. User selects Book option
4. System shows Book entry form
5. User fills Book entry form with Book data
6. System saves Book

**UI Elements**

Select entry type dialog
- show
- Book option
  - show
  - select

Book entry form
- save
- Book
  - Author
  - Title

**Entities**

Book data

**Domain vocabulary**

«invoke» Delete entry
Recovery

Automated recovery

Manual recovery

Interface layer
- GUI
- services

Application logic
- Flow control

Business logic
- Algorithms
- Data
TALE tool

1. User selects **Nowa sprawa kredytowa** from Sprawy kredytowe menu
2. System shows **Rejestrowanie nowej sprawy kredytowej window**
3. User enters **Rejestrowanie nowej sprawy kredytowej data**
Migration to modern technology system
Solution overview

public VCourseForm() {
  /* ... */
  jbnClickHere = new JButton("Click Here");
  this.add(jbnClickHere);
  jbnOK.addActionListener(
    /* ... */
    jbnClickHereActionPerformed(evt);
    /* ... */
  );
}

private void jbnClickHereActionPerformed(java.awt.event.ActionEvent evt) {
  cAddNewCourse.clicksClickHereButton();
  /* ... */
}

Copyright © 2012

GUI-ripping tool

TALE+ReDSeeDS

Code

VSomeFormClass

User clicks "Click here" button
3-layer architecture

Presentation

Controller

Model

MCourse

MCourseList

CAddNewCourse

CShowOwnedCourseList

VCourseForm

VCourseListForm

ErrorMessage

VMCourseForm

VMCourseListForm

VErrorMessage

Presentation

Controller

Model

MCourse

MCourseList

CAddNewCourse

CShowOwnedCourseList

VCourseForm

VCourseListForm

ErrorMessage

VMCourseForm

VMCourseListForm

VErrorMessage
MOLA

- **Model transformation Language**
- **Graphical language**
- **Based on metamodel-level pattern matching**
- **Transformation (program) describes how to transform one model into another**
MOLA - example
Summary

- Combination of existing approaches:
  - Software Reverse Engineering
  - Requirements Engineering
  - Model Driven Development

- Shortens time needed for transition from legacy to modern technology system

- Validated using a real-life system
Thank You!