The GUISurfer tool

GUI Inspection from Source Code Analysis

João Carlos Silva[1,2] João Saraiva[1] José Creissac[1]

[1] Departamento de Informática/CCTC
Universidade do Minho
[2] Departamento de Tecnologia
Instituto Politécnico do Cávado e do Ave

Portugal

Fourth International Workshop on Foundations and Techniques for Open Source Software Certification
(OpenCert 2010)
Setember 17-18, 2010

GUI Model-based Analysis

Models guide the development process
Models help in the maintenance and evolution of software

Objectives:

- Static analysis of interactive applications from source code
  - Usability and implementation quality
- GUI maintenance and evolution

Approach:

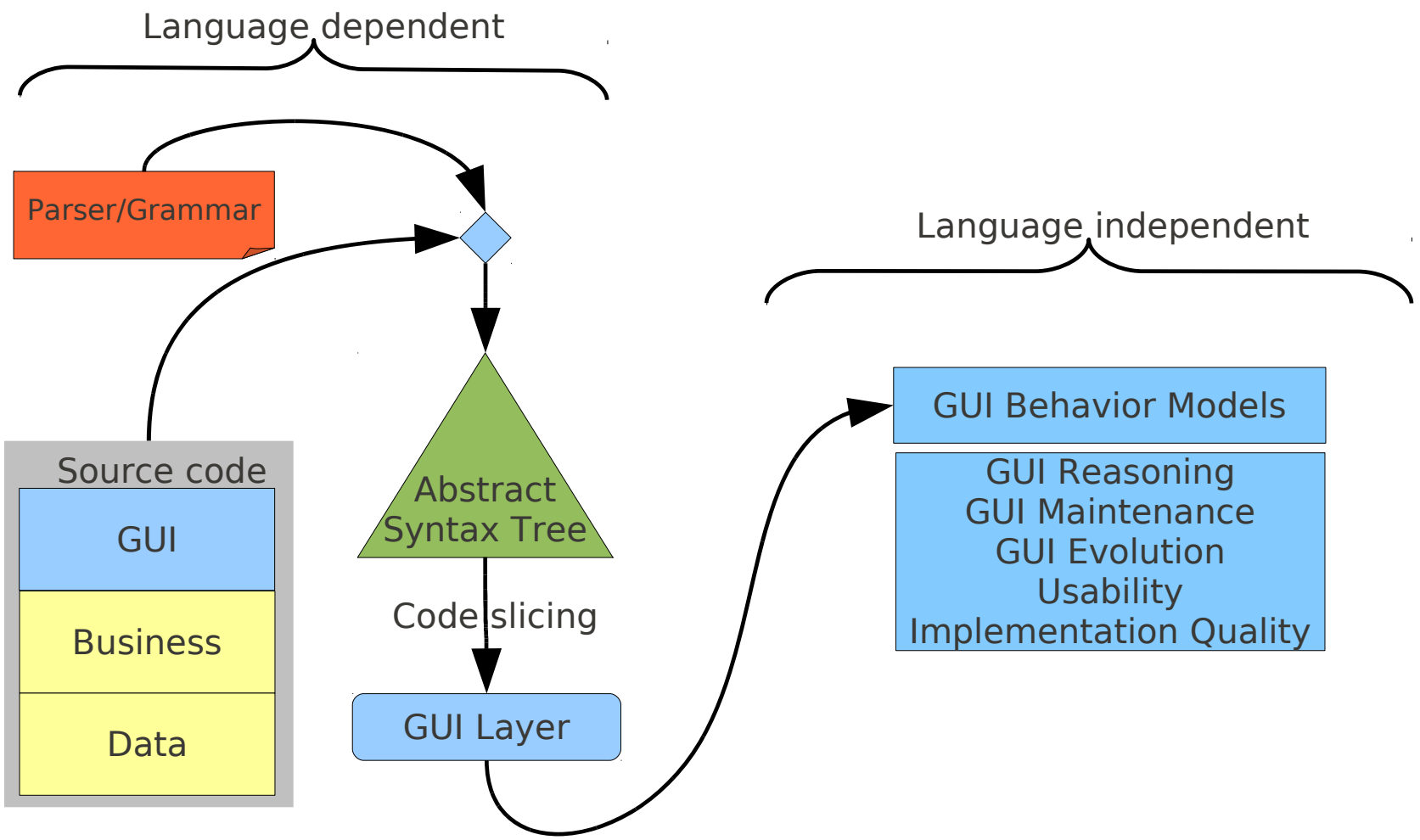
- Extraction of GUI layer (event-based toolkits)
- Automatic extraction of behavioral models

Paper Goal

On going work:

- Models extraction from source code
- GUI behavior representation
- Integration of testing techniques
- Language-independent tool

The GUISurfer Architecture



Reverse Engineering of GUI source code

Generic approach easily re-targetable:

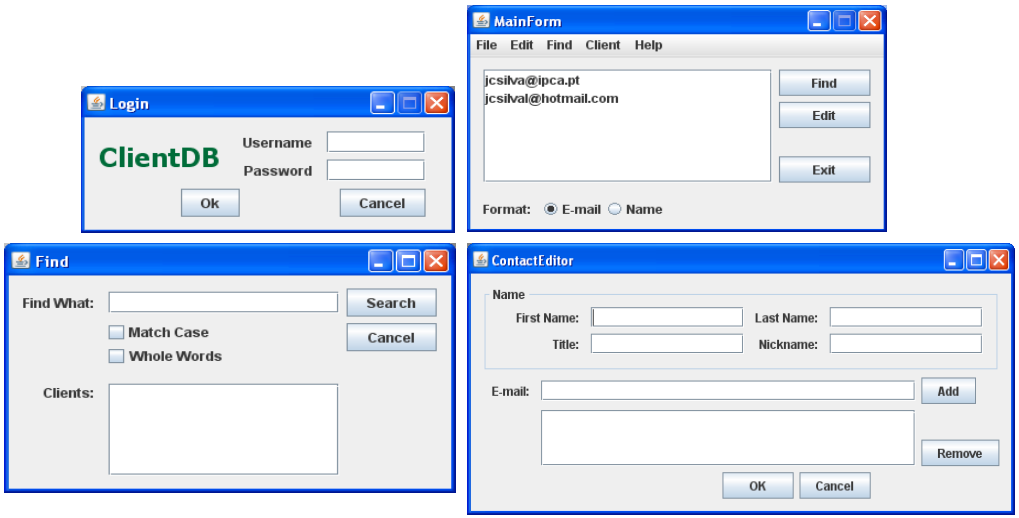
- Look for widgets: user input, user selection, user action, output to user
- Traversal of the AST to detect GUI subtrees
- Based on program dependency graph
- Pre-defined set of generic traversal functions
- GUI components constructors

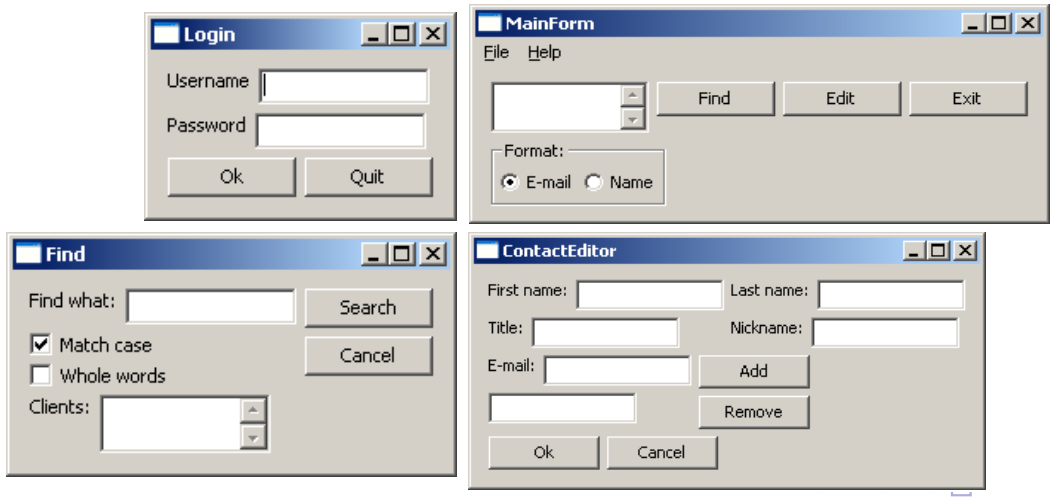
GUISurfer - Configurable tools

GUISurfer is composed by three tools:

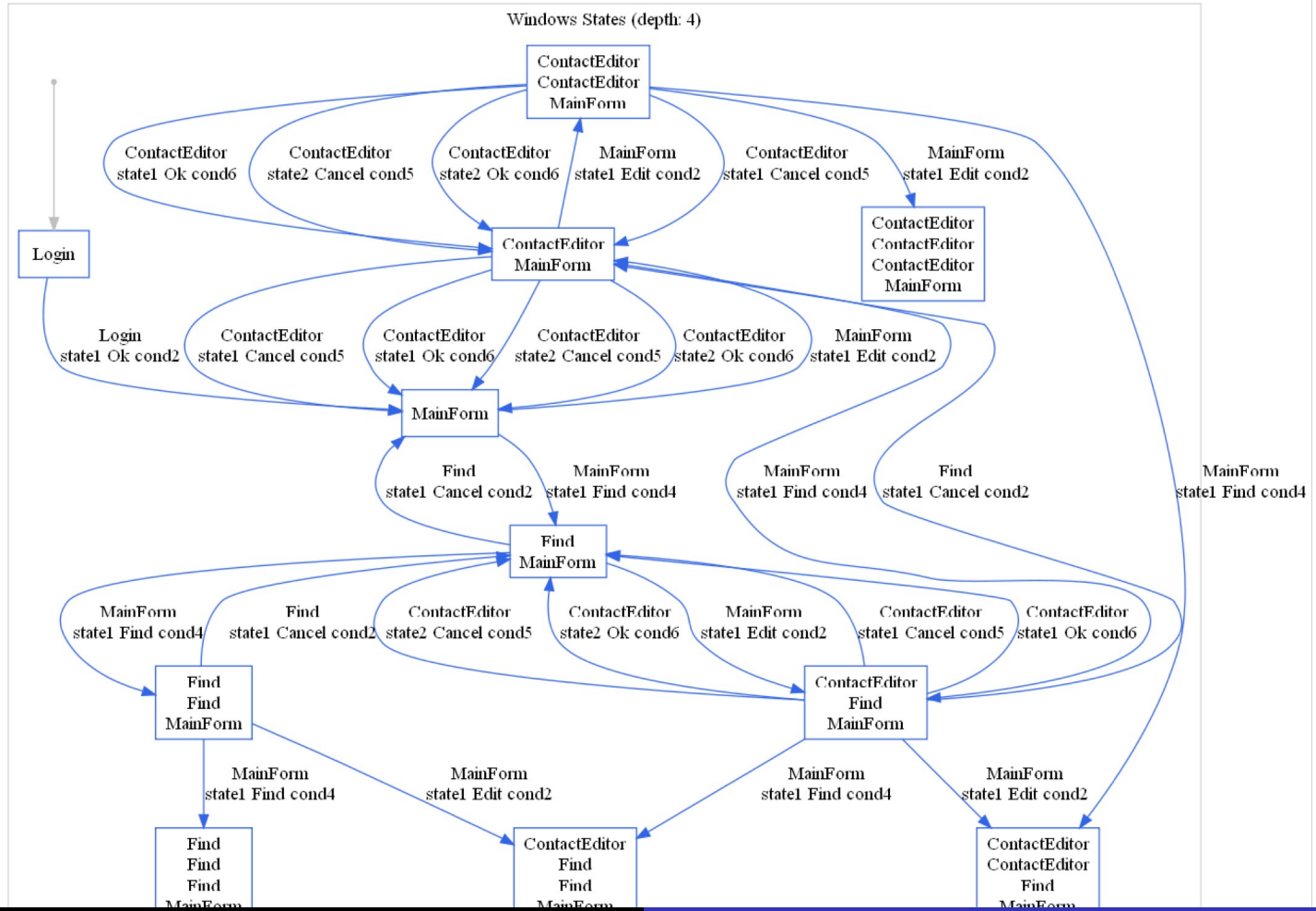
- FileParser: Parse a particular source code file. Generate the abstract syntax tree
- AstAnalyser: Slice an abstract syntax tree. Entry point and list of widgets as parameters
- Graph: Generates several metadata files and models with events, conditions, actions and states (Set of fragments of the AST)

An Interactive Agenda - Java/Swing - WxHaskell



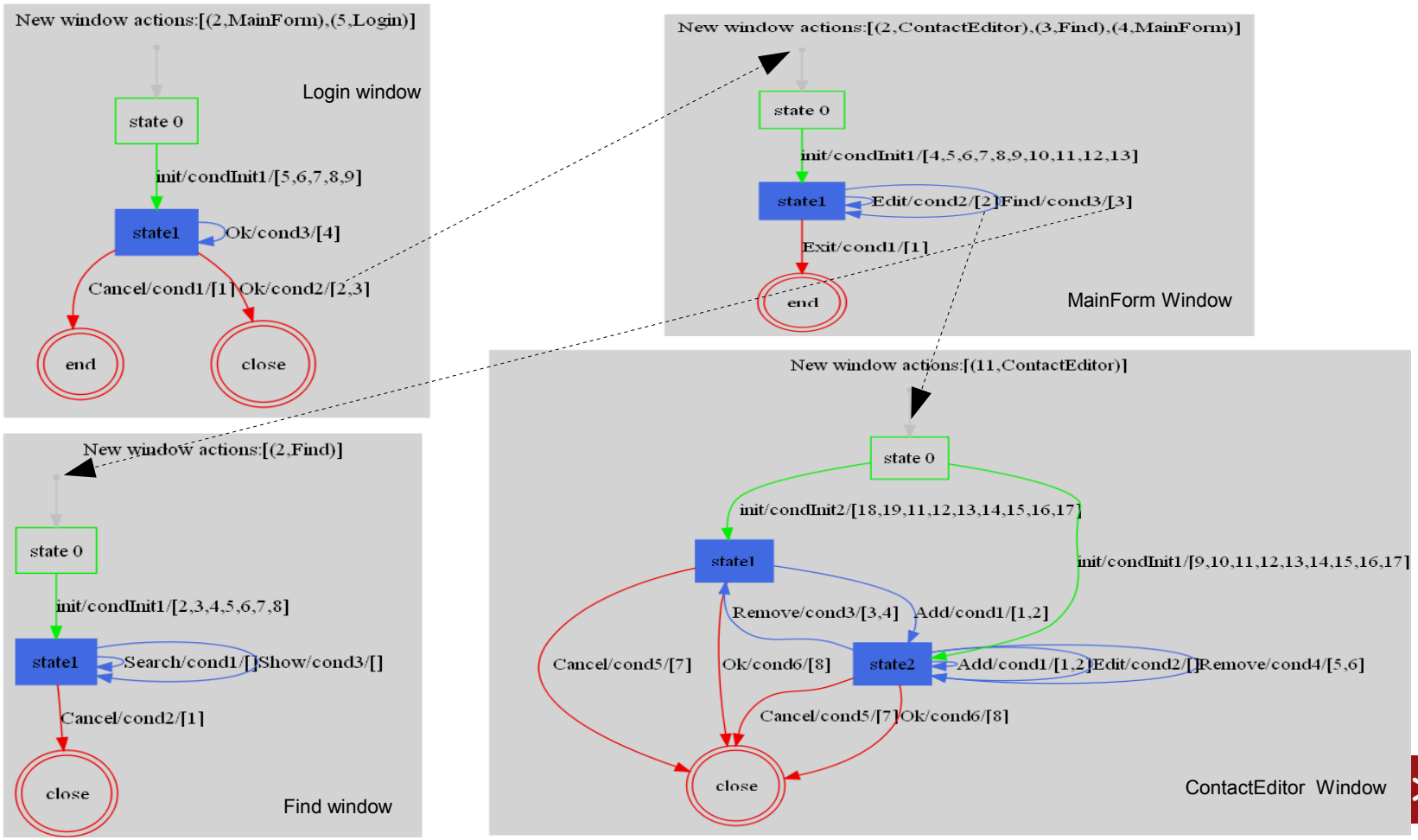


Flow Between Windows



Window Internal Behavior

Windows behavior:

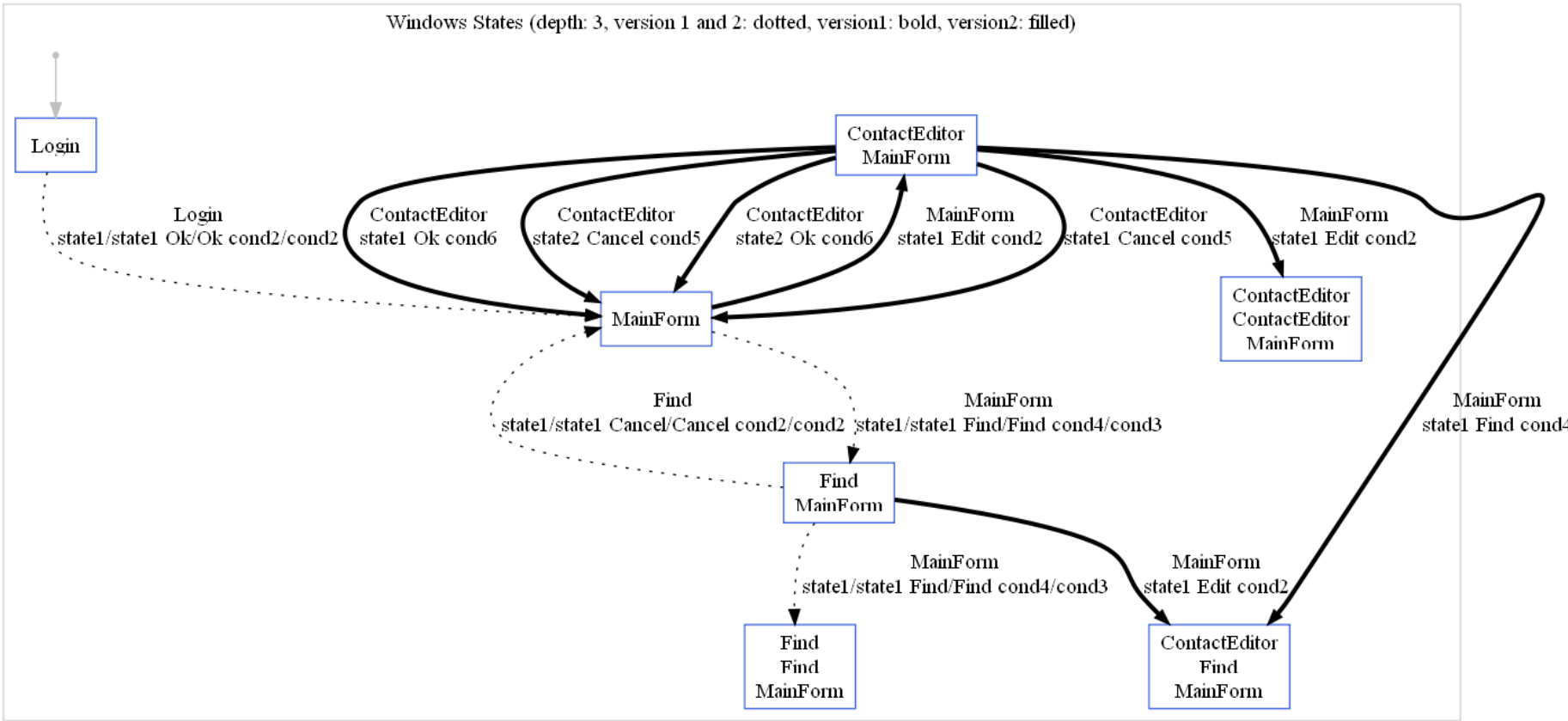


Graph Theory Applied to State Machine Reasonning

- States machines as graphs: States are vertices, events are edges
  - Refactoring: We can compute an equivalent machine with the minimum number of states
  - Dead code elimination: Detect if all states are reachable from the initial one
  - Test case generation:
    - Chinese Postman algorithm checking every event through properties
    - Traveling salesman checking every state through properties
  - Center of a graph: main window at a central point

Graph Operators: Intersection, Union and Difference

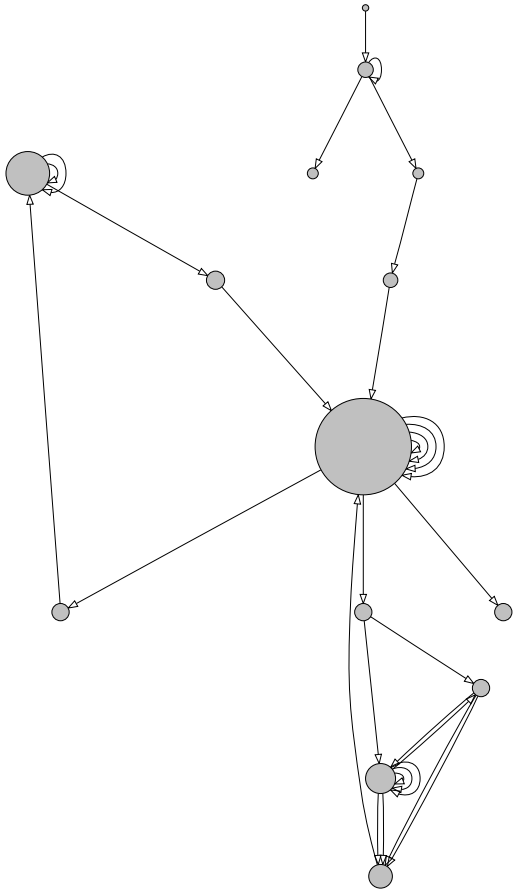
Intersection of two different versions



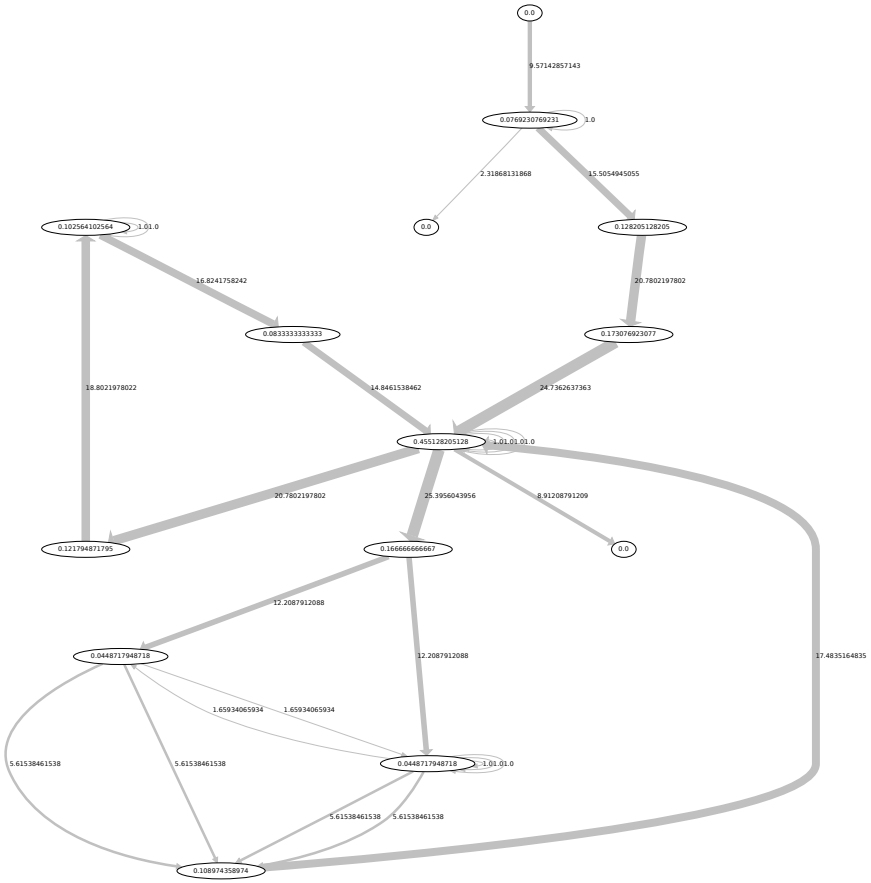
GUI Metrics

- Use of Graph-Tool for the manipulation and statistical analysis of graphs;
- Shortest distance between vertices;
- Pagerank;
- Betweeness;
- Cyclomatic complexity;

Pagerank Algorithm



Betweenness Algorithm



GUI Test Cases Generation

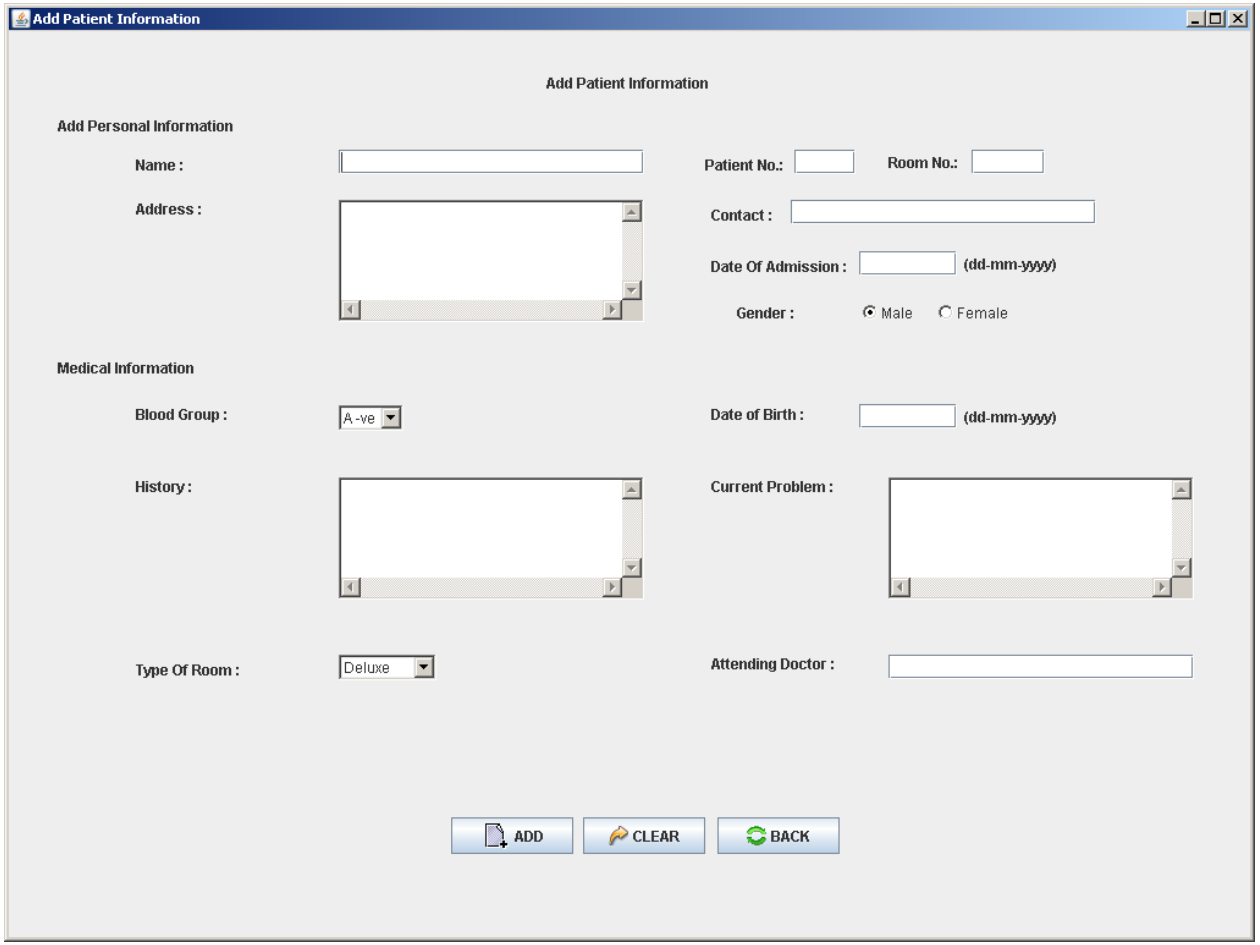
- Coverage criteria:
  - event coverage: set of event-sequences considering all possible events;
  - state coverage: each state reached at least once;
  - length-n-event-sequence coverage: set of event-sequences which contains all event-sequences of length equal to n;
- Chinese Postman Tour;
- Travelling Salesman Problem;

Case Study

Java/Swing application: Healthcare management system

- source code from Planet-Source-Code.com
- 66 classes;
- 29 windows forms (included message box);
- patients and doctors management;
- bills of patients display;

Windows Form Example: Add Patient



Case Study: Results

...

Ongoing Work

- Use of *QuickCheck*: a *haskell* library tool for testing *haskell* programs automatically.
  - *haskell* specification of the GUI
  - Properties definition
  - QuickCheck tests the properties in a large number of randomly generated cases
- GUISurfer back-end extension: generalizing the approach to new languages and toolkits (GWT, JavaScript)

Conclusions and Future Work

- Able to derive user interface models of interactive computing system
- Support for Java, WxHaskell and GWT available
- Plan to extend our implementation to handle more complex user interfaces
- Development of a web portal

The GUISurfer tool

GUI Inspection from Source Code Analysis

João Carlos Silva[1,2] João Saraiva[1] José Creissac[1]

[1] Departamento de Informática/CCTC
Universidade do Minho
[2] Departamento de Tecnologia
Instituto Politécnico do Cávado e do Ave

Portugal

Fourth International Workshop on Foundations and Techniques for Open Source Software Certification
(OpenCert 2010)
Setember 17-18, 2010