Imperial College London

Refactoring Raptor

Group Project

Spyridon Karavanis	sk6109
Andrew Morton	am12709
Vivek Sangar	vs309
Anastasios Karavanis	ak8809
Alexandre Krol	ak5409

Supervisor: Dr. Krysia Broda

Outline

- Motivation
  - What was the initial problem?
- Methodology
  - How we approached it?
- Outcome
  - What is the final product?

Imperial College London

Motivation

- RAPTOR
  - Tool for Reasoning About Programs
  - Developed by 1st and 2nd year Students
- Original System
  - Lack of OO Design Principles
  - Lack of Unit Testing
  - Lack of Exception Handling Structure

Project Requirements (i)

- Must-have Requirements:

  - ✓Implementation of the MVC design pattern
  - ✓Saving Proofs in XML files
  - ✓Unit Test Suite
  - ✓Improve Exception Handling

Imperial College London

Project Requirements (ii)

- Should-have Requirements:

  - ✓Fix the Help System
  - ✓Platform Independence (support JRE 1.6)
  - ✓Fix Save File Functionality
  - ✓Fix Bugs

Project Requirements (iii)

- Could-have Requirements:

  ✓ Embed the Help System in a Wiki

  - Saving the Model in a XML file

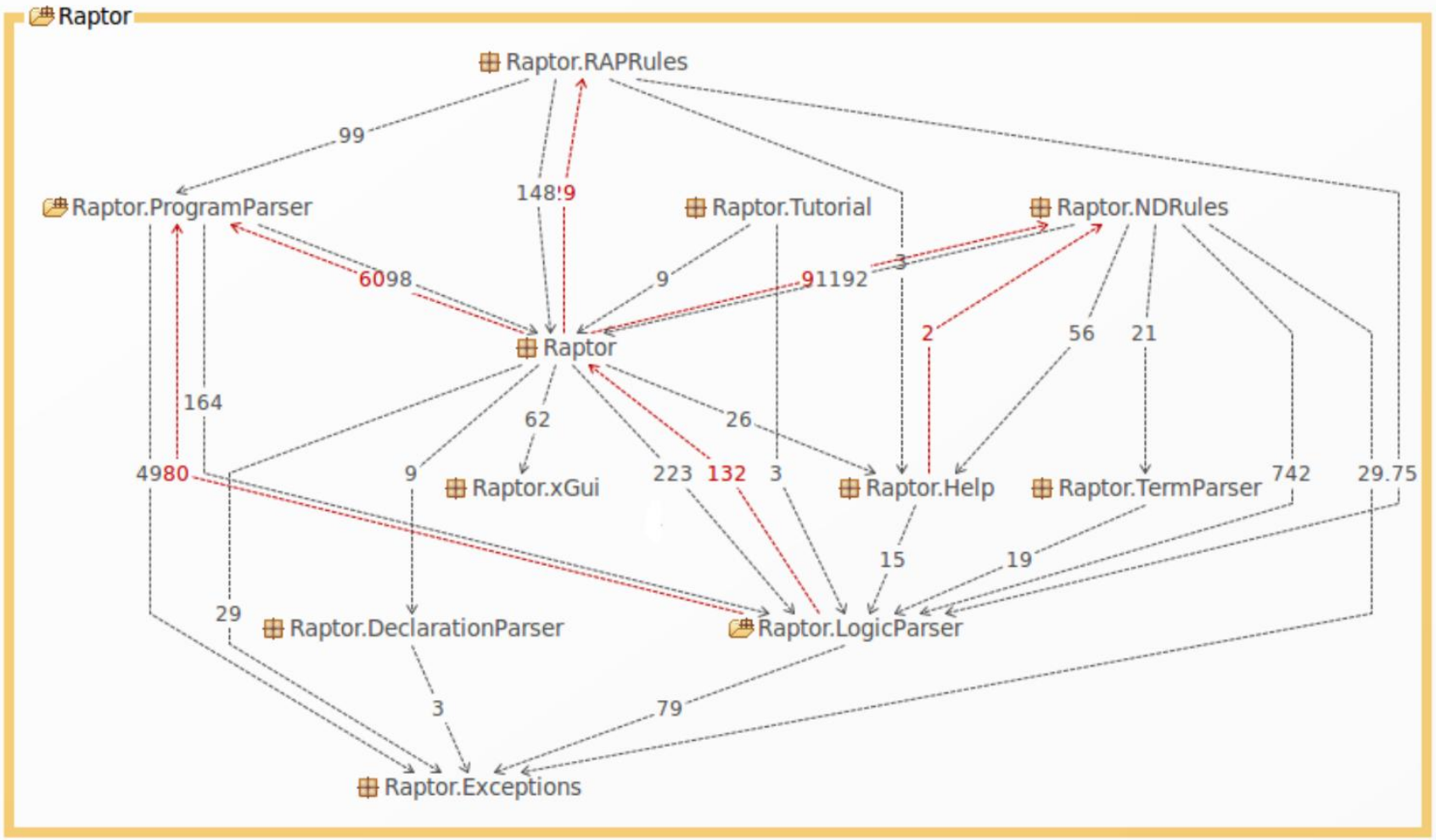
  ✓ Improve the GUI – Usability

  × Improve Boolean/method Handling

  × Improve Parser Error Handling

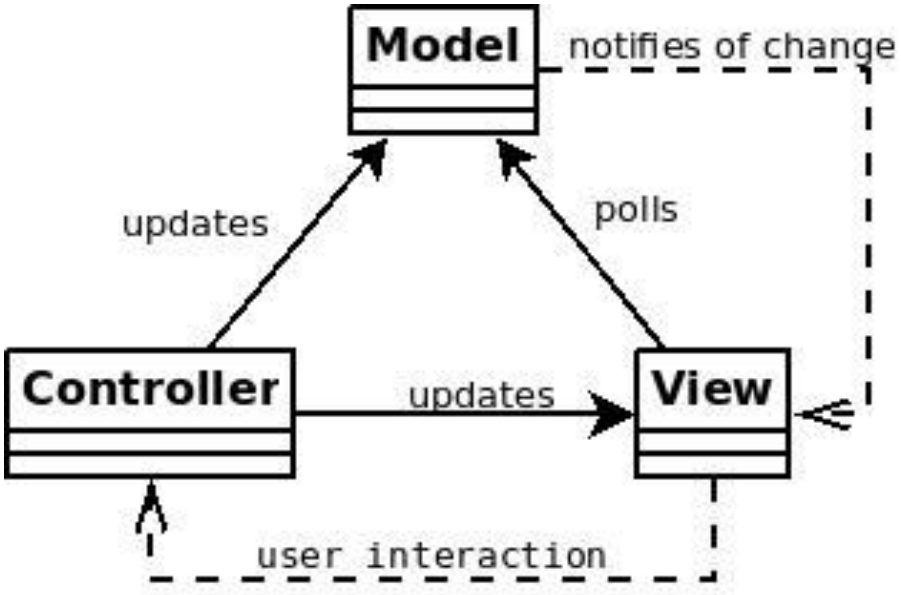
Imperial College London

Initial Architecture



MVC Architecture

- Model-View-Controller:
  - **Model**: data and business logic
  - **View**: displays model
  - **Controller**: performs operations on the model



Observer Pattern

- **Observable** entity viewed by many **Observables**
- **Observable** notifies observers of change
- **Observers** conform to well defined interface

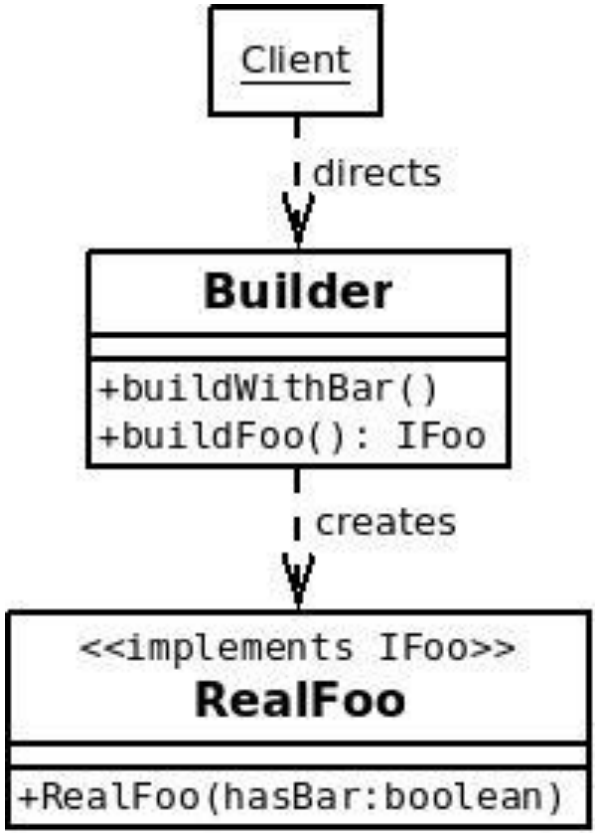
Dependency Injection

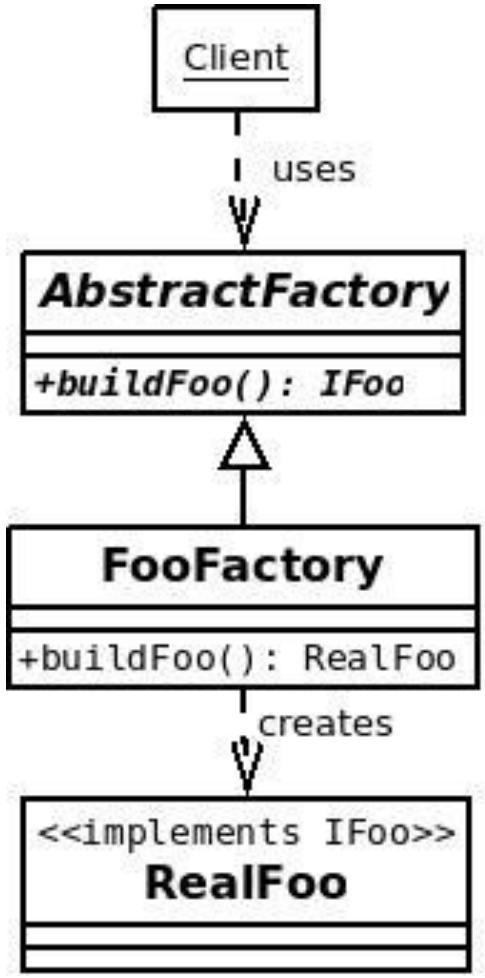
- Eliminate use of the word *new* in business logic code
- All things required by a class given in constructor/ methods
- Allows easier:
  - Testing
  - Swapping out of components
- Disadvantage – Long constructors

Builder Pattern

- Have another object that controls object creation
- Calling class doesn't know what concrete class it is receiving
- Differences:
  - Builders build objects step by step
  - Abstract factories build object using other factories

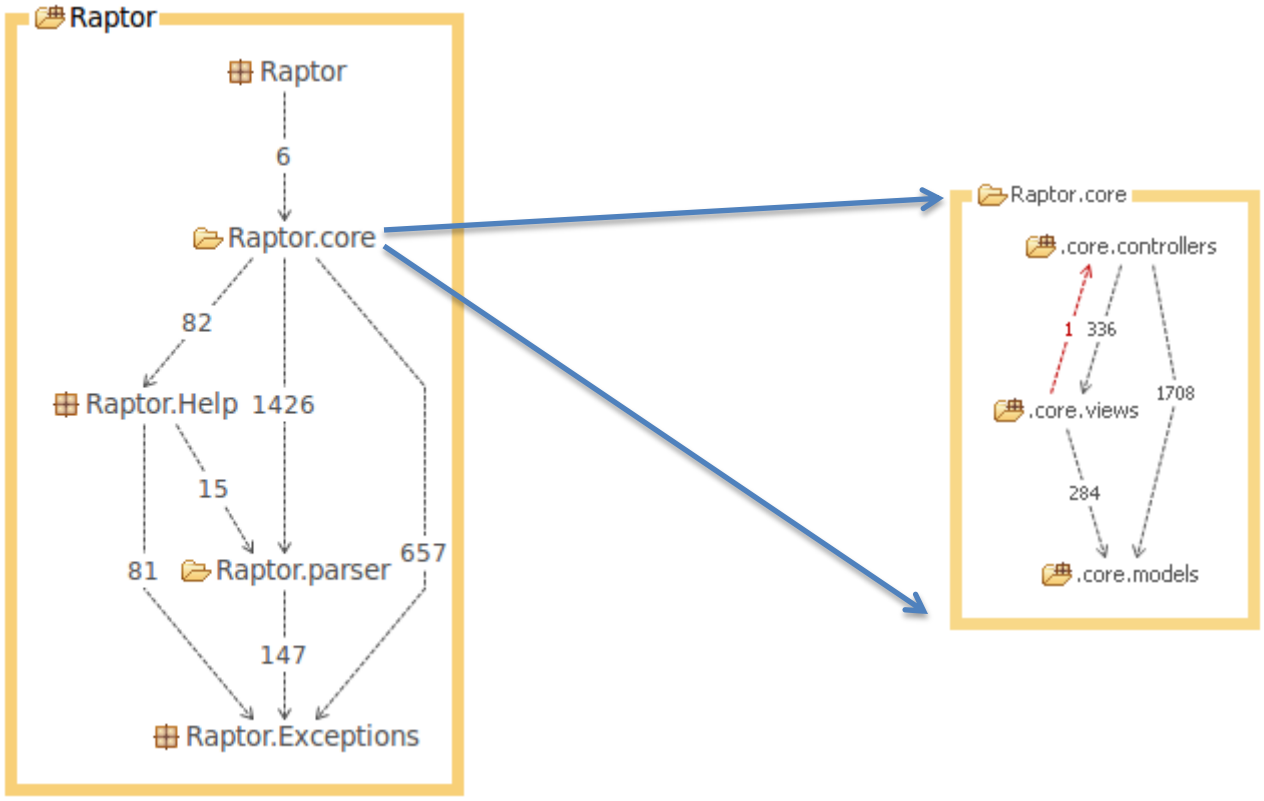
Builder/Factory Pattern





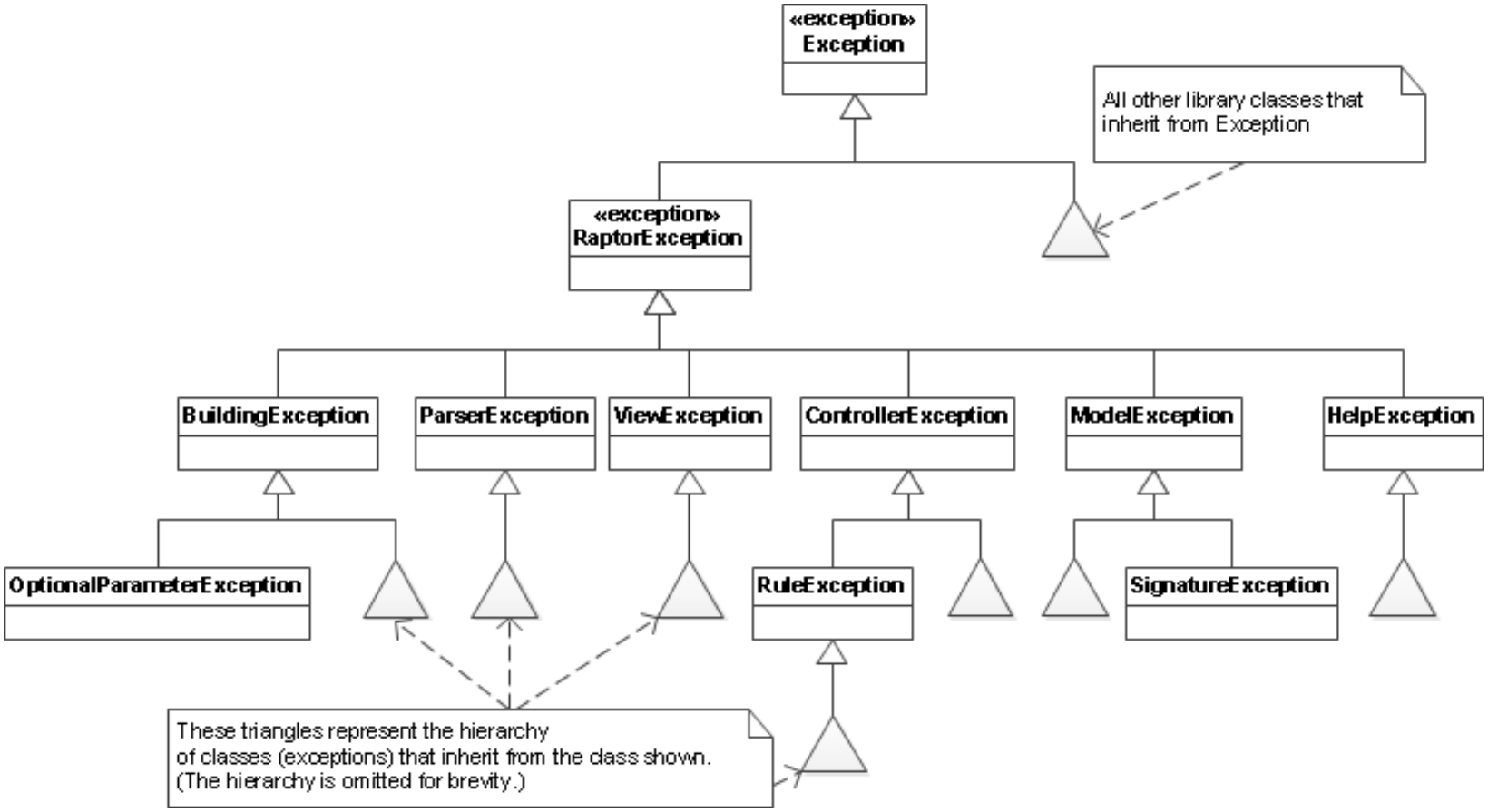
Imperial College London

New Architecture



Exception Handling

- Tree-Hierarchy



XML Storage of Proofs (i)

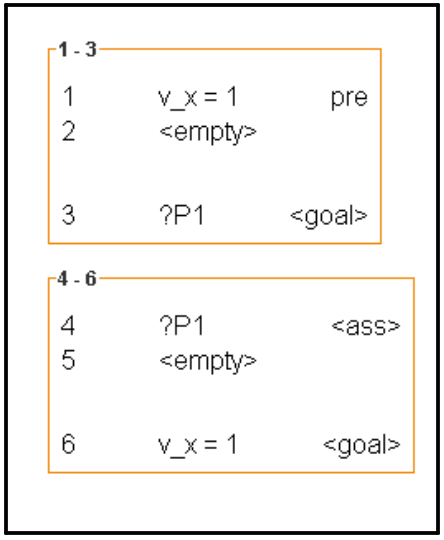
- XStream library
  - Serializing **Model** Components

```
…
XStream xstream = new XStream();
ObjectOutputStream out =
xstream.createObjectOutputStream(oos);

out.writeUTF(pre.getText());
…
if (super.inProofMode()) {
        out.writeObject(getMainProofBox());
        …
}
```

XML Storage of Proofs (ii)

- XML Export
  - Export only Proofbox
  - Future work: XML Schema definition



Export

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <raptor>
  - <box>
    - <box>
        <line line_nb="1" just="pre">v_x = 1</line>
        <line line_nb="2" just="">{<empty>}</line>
        <line line_nb="3" just="{<goal>}">?P1</line>
      </box>
    - <box>
        <line line_nb="4" just="<ass>">?P1</line>
        <line line_nb="5" just="">{<empty>}</line>
        <line line_nb="6" just="{<goal>}">v_x = 1</line>
      </box>
    </box>
  </raptor>
```

Unit Tests (i)

- Why unit test?
  - Automated tests prove features
  - Prove the software still works (as tested).
  - Confidence, quality
- Techniques:
  - JUnit: Framework for unit testing in Java
  - EclEmma: Test code coverage tool for Eclipse
  - jMock: Framework to simulate (mock) objects

Imperial College London

Unit Tests (ii)

- What we have done:
  - Initial version did not have any tests
  - Current version has 13.1% test coverage
  - Few tests about what we didn't change

Element	Coverage	Covered Instructions
Raptor	13,1 %	10644
src	13,1 %	10644
Raptor.core.controllers.building.rules.nd	100,0 %	694
Raptor.core.models.building	35,2 %	597
Raptor.parser.LogicParser	0,1 %	4
Raptor.core.controllers.building.rules	90,9 %	130
Raptor.core.controllers.building.rules.rap	100,0 %	131
Raptor.test.core.controllers.RAPRules	100,0 %	152

Unit Tests (iii)

- JUnit TestSuite:
  - Run test cases in only one execution
  - Easy to add new tests

```
public class RaptorTestSuite {
    public static Test suite() {
        TestSuite suite = new TestSuite();
        // Here follows all the TestCases
        suite.addTestSuite(RaptorTest.class);
        // ...
        return suite;
    }
    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }
}
```

Unit Tests (iv)

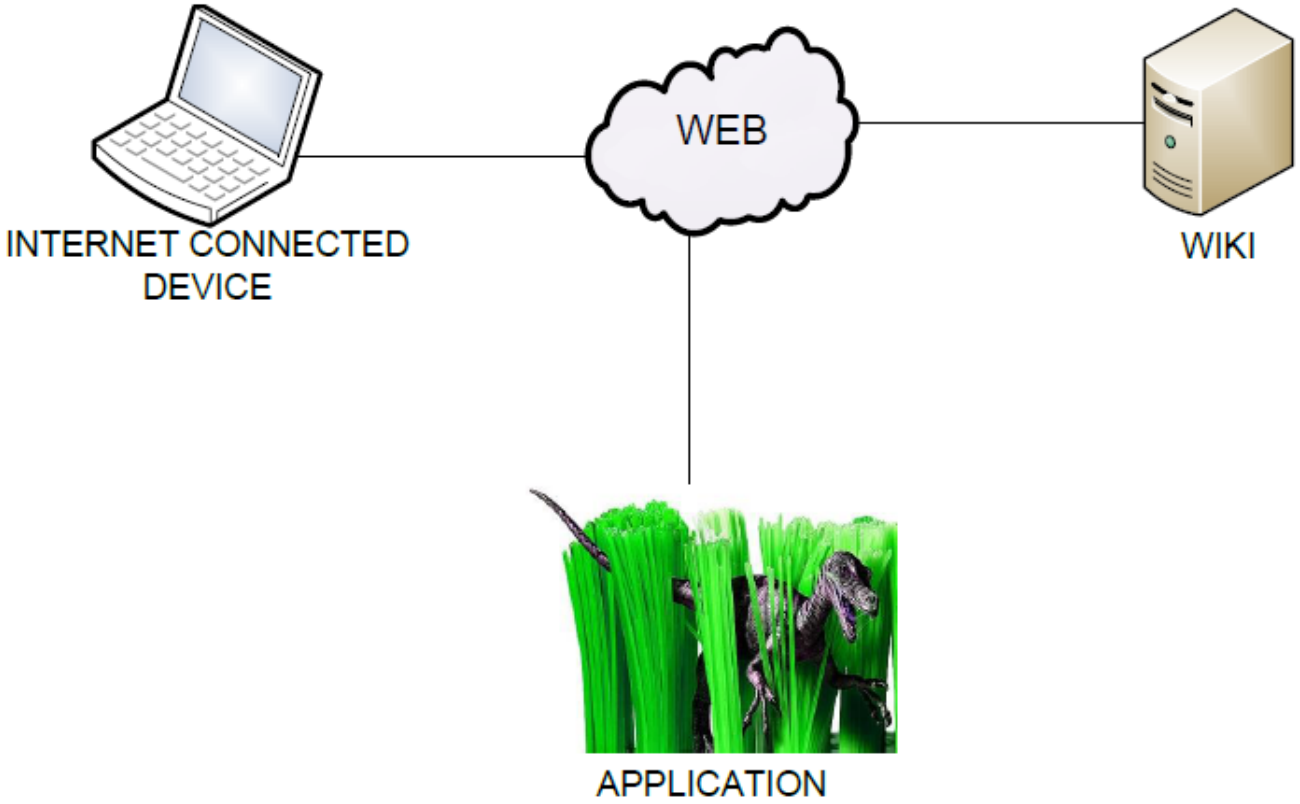
- jMock:
  - Simulate objects for test
  - A good solution to use interfaces without using implemented object

```
public void testCreate() {
    RuleControllerFactory controllerFactory =
         new RuleControllerFactory(iRuleFactoryManager);
    context.checking(new Expectations() {
      {
        oneOf(iRuleController).setProofBox(proofBoxMock);
        oneOf(iRuleController).setUser(userMock);
      }
    });
    controllerFactory.with(proofBoxMock);
    controllerFactory.with(userMock);
    assertTrue(controllerFactory.create() instanceof IRuleController);
}
```

Imperial College London

Help System in Wiki

- MediaWiki Distribution
- Setup on Departmental Server



Imperial College London

Help System

- Use of Properties Files

```
private static Properties tempProp = new Properties();
private static final URL exceptions =
ClassLoader.getSystemResource("……/exceptions.properties");

public static String getException(String exception) {
        ....
        if (tempProp.isEmpty())
        tempProp.load(exceptions.openStream());
        ......
        return tempProp.getProperty(exception);
}
```

```
###### Exceptions Messages #####
ruleController = Please select an empty or goal line.
selectLine = You have to select a proof line.
```

Other Improvements

- Fixed Bugs
  - Save File Functionality
  - Export to Latex
  - etc.
- Platform Independence
  - JRE 1.6
- GUI Improvement
  - External Libraries
  - Swing Components

Software Engineering Techniques

- SCRUM
  - Short Iterations
  - Product Backlog
  - Sprint Backlog
- XP
  - Communication
  - Feedback
  - Simplicity

Conclusion

- Challenge Of Project
- Project Goal Completed
- Project Followed Initial Schedule
  - 80-90% was done according to schedule
- Advantages Gained
  - Extensibility
  - Simplicity
  - Generic

Imperial College London

Questions

Thanks For Your Attention