Test Automation at the useR interface level

Tanja E. J. Vos
Software Testing and Quality Group
Research center for Software Production Methods (PROS)
Universidad Politecnica de Valencia
Spain

SATToSE 2014, L’Aquila 2014
Contents

• FITTEST project
• Testing at the UI level: what and state-of-the-art
• TESTAR and how it works
• How it has been used
FITTEST

- Future Internet Testing
- September 2010 – February 2014
- Total costs: 5.845.000 euros
- Partners:
  - Universidad Politecnica de Valencia (Spain)
  - University College London (United Kingdom)
  - Berner & Mattner (Germany)
  - IBM (Israel)
  - Fondazione Bruno Kessler (Italy)
  - Universiteit Utrecht (The Netherlands)
  - Softteam (France)
- http://www.pros.upv.es/fittest/
FITTEST objectives/results

• Future Internet Applications
  – Characterized by an extreme high level of dynamism
  – Adaptation to usage context (context awareness)
  – Dynamic discovery and composition of services
  – Limited observability (3rd party black-box components)
  – Etc..

• Testing of these applications gets extremely important
  • Society depends more and more on them
  • Critical activities such as social services, learning, finance, business.

• Traditional testing is not enough
  – Testwares are fixed

• Continuous testing is needed
  – Testwares that automatically adapt to the dynamic behavior of the Future Internet application
  – This is the objective of FITTEST
The FITTEST tools for Continuous Testing
FITTEST continuous testing system

LOGGING

1. Run the target System that is Under Test (SUT)
2. Collect the logs it generates

This can be done by:

- real usage by end users of the application in the production environment
- test case execution in the test environment.
How does it work?

1. Analyse the logs
2. Generate different testwares:
   - Models
   - Domain Input Specification
   - Oracles
3. Use these to generate and automate a test suite consisting of:
   - Abstract test cases
   - Concrete test cases
   - Pass/Fail Evaluation criteria
How does it work?

Execute the test cases and start a new test cycle for continuous testing and adaptation of the test wares!
And it does work, but....... 

• We cannot always get the logs...

• The logs do not always contain the info we need to derive a good model/oracle

• Instrumentation is not always an option (3rd party components)
Do you want to know more...

• Vos, Tanja E.J., Lakhotia, Kiran, Bauersfeld, Sebastian (Eds.) *Future Internet Testing*, LNCS 8432, 2014

• Paolo Tonella youtube lecture: [https://www.youtube.com/watch?v=TnuiEGS6iyC](https://www.youtube.com/watch?v=TnuiEGS6iyC)

If we cannot rely on the logs, why not rely on what we can see.... the UI
Testing at the UI Level

• UI is where all functionality comes together
  – Integration / System Testing

• Most applications have UIs
  – Computers, tables, smartphones....

• Faults that arise at UI level are important
  – These are what your client finds -> test from their perspective!

• No need for source code
  – But if we have it even better ;-)
State of the art in UI testing

- **Capture Replay**
  - The tool captures user interaction with the UI and records a script that can be automatically replayed during regression testing
  - UI change (at development time & at run time)
  - Automated regression tests break
  - Huge maintenance problem

- **Visual Testing**

- **Model-based Testing**
State of the art in UI testing

- **Capture Replay**

- **Visual testing**
  - Based on image recognition
  - Easy to understand, no programming skills needed
  - Solves most of maintenance problem
  - Introduces additional problems:
    - Performance of image processing
    - False positives and false negatives
      - the ambiguity associated with image locators
      - imprecision of image recognition feeds into oracles

- **Model-based Testing**
State of the art in UI testing

- Capture Replay
- Visual testing

- (ui) Model-based testing -- TESTAR
  - Based on automatically inferred tree model of the UI
  - Tests sequences are derived automatically from the model
  - Executed sequences can be replayed
  - If UI changes so does the model/tests -> no maintenance of the tests
  - Programming skills are needed to define powerful oracles
    - It needs to be investigated more if this is really a problem....
    - Do we want testers to have programming skills?
How it works..

START SUT → SCAN GUI + OBTAIN WIDGET TREE → DERIVE SET OF USER ACTIONS

Optional instrumentation

more sequences?

calculate fitness of test sequence

more actions?

FAULT?

SELECT ACTION

EXECUTE ACTION

ORACLE

STOP SUT

Replayable Erroneous Sequences

Domain Experts

Action Definitions

Oracle Definition
START SUT

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

START SUT

optional instrumentation

more sequences?

STOP SUT

fitness sequence

- Run executable / command
- Bring SUT into dedicated start state (delete or restore configuration files)
- Wait until SUT fully loaded

FAULT?

Yes

No

ORACLE

SELECT ACTION

- Run executable / command
- Bring SUT into dedicated start state (delete or restore configuration files)
- Wait until SUT fully loaded

ORACLE

possible fault?

more sequences?

STOP SUT

optional instrumentation

SUT

Domain Experts

Action Definitions

Oracle Definition

Replayable Erroneous Sequences
SUT

Domain Experts

optional instrumentation

Obtain state (**Widget Tree**) through Accessibility API

START SUT

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

Action Definitions

Oracle Definition

Oracle Fault?

SELECT ACTION

EXECUTE ACTION

STOP SUT

calculate fitness of test sequence

more actions?

FAULT?

Replayable Erroneous Sequences

ORACLE

No

Yes

No

Yes

STOP SUT
Widget Trees

Example

File   Edit   View   Help

This is a multiline text.

Example

Button

This is a multiline text.

ABC

Window

Button

Text

Menu

Slider

MI

MI

MI

type: TMenuItem

title: "File"

...
Active Widget Tree

Form

Main Menu

(disabled)

Search
An outline is not available.

Create CTE Diagram
Specify file name for new CTE file.

Files:
C:\(Documents and Settings)\(bern&mattner\)default.cte

Finish  Cancel

Dialog
- Use information in Widget Tree to derive a set of “sensible” actions
- Click on enabled Buttons, Type into Text Boxes...
- Select one of the actions from the action set
- Various possible strategies:
  - Random,
  - Coverage Metrics,
  - Search-based...
START SUT ➔ SCAN GUI + OBTAIN WIDGET TREE ➔ DERIVE SET OF USER ACTIONS

SUT

more sequences?

optional instrumentation

STOP SUT ➔ calculate fitness of test sequence ➔ more actions?

Execute and record selected action

EXECUTE ACTION ➔ ORACLE

Replayable Erroneous Sequences

Domain Experts ➔ Action Definitions ➔ Oracle Definition

START SUT ➔ SCAN GUI + OBTAIN WIDGET TREE ➔ DERIVE SET OF USER ACTIONS

more sequences?

calculate fitness of test sequence ➔ more actions?

fault?

execute and record selected action

execute action ➔ oracle

more sequences?

optional instrumentation

replayable erroneous sequences

domain experts ➔ action definitions ➔ oracle definition
START SUT

optional instrumentation

more sequences?

STOP SUT

calculate fitness of test sequence

more actions?

Oracle -> Check whether state is erroneous

scan gui + obtain widget tree

derive set of user actions

select action

execute action

fault?

oracle definition

action definitions

domain experts

replayable erroneous sequences
Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?

more actions?

No

Yes

Stopping Criteria:
- After X actions
- After Y hours
- After some state occurred
- etc....

Did we find a fault?

Replayable Erroneous Sequences

SCAN GUI + OBTAIN WIDGET TREE

DERIVE SET OF USER ACTIONS

SELECT ACTION

EXECUTE ACTION

ORACLE

FAULT?
Domain Experts

START SUT

START SUT

optional instrumentation

Save sequence if it contained errors

more sequences?

STOP SUT

STOP SUT

more actions?

sequence

Yes

Yes

FAULT?

No

Yes

EXECUTE ACTION

SELECT ACTION

DERIVE SET OF USER ACTIONS

SCAN GUI + OBTAIN WIDGET TREE

more actions?

Calculate fitness of test sequence

No

Oracle Definition

Oracle Definition

Action Definitions

Replayable Erroneous Sequences
START SUT

optional instrumentation

more sequences?

STOP SUT

calculate fitness of test sequence

No

more actions?

No

FAULT?

Yes

EXECUTE ACTION

SELECT ACTION

DERIVE SET OF USER ACTIONS

SCAN GUI + OBTAIN WIDGET TREE

ACTION Definitions

Oracle Definition

Domain Experts

SUT

Replayable Erroneous Sequences

Various stopping criteria:
- X sequences
- Y hours
- ...

TEST*
TESTAR tool

READY

Set path the SUT
Filter:

1) undesirable actions, i.e. closing the application all the time

2) Undesirable processes, for example help panes in acrobat, etc......
GO!

See video at https://www.youtube.com/watch?v=PBs9jF_pLCs
Oracles for free

• What can we easily detect?
• Crashes

• Program freezes
Cheap Oracles

- Critical message boxes
- Suspicious stdout / stderr
Specifying Cheap Oracles

- Simply with regular Expressions
- For example:
  - .*NullPointerException
  - .*\([Ee]rror\)|\([Pp]roblem\)
More sophistication needs work

• Actions
  – Action detection
  – Action selection
  – Sometimes a trial/error process
    • Random selection = like a child, just much faster
    • Printing, file copying / moving / deleting
    • Starts other Processes
    • Rights management, dedicated user accounts, disallow actions

• Oracles that need programming
How? Edit the protocol
The protocol editor
protected SUT startsSystem() throws SystemStartException
protected State getState(SUT system)
throws StateBuildException
protected Set<Action> deriveActions(SUT system, State state) throws ActionBuildException
protected Action selectAction(State state, 
Set<Action> actions);

// Here you can implement any selection strategy
// per defaults this is random selection from actions
protected boolean executeAction(SUT system, State state, Action action);
protected Verdict getVerdict(State state);
protected Verdict getVerdict(State state){
  Assert.notNull(state);

  //-----------------------
  // ORACLES FOR FREE
  //-----------------------

  // if the SUT is not running, we assume it crashed
  if(!state.get(IsRunning, false))
    return new Verdict(1.0, "System is offline! I assume it crashed!");

  // if the SUT does not respond within a given amount of time, we assume it crashed
  if(state.get(NotResponding, false))
    return new Verdict(0.8, "System is unresponsive! I assume something is wrong!");

public final class Verdict {
  ....
  private final String info;
  private final double severity;
  private final Visualizer visualizer;

  public Verdict (double severity, String info)
  public Verdict(double severity, String info, Visualizer v)
String titleRegEx = settings().get(SuspiciousTitles);

// search all widgets for suspicious titles
for(Widget w : state){
    String title = w.get(Title, "") ;
    if(title.matches(titleRegEx)){
        // visualize the problematic widget, by marking it with a red box
        Visualizer visualizer = Util.NullVisualizer;
        if(w.get(Tags.Shape, null) != null){
            Pen redPen = Pen.newPen().setColor(Color.Red).(...).build();
            visualizer = new ShapeVisualizer(redPen, ..... , "Suspicious Title", 0.5, 0.5);
        }
        return new Verdict(1.0, "Discovered suspicious widget title: " + title + ".", visualizer);
    }
}
getVerdict

//---------------------------------------------------------------
// MORE SOPHISTICATED ORACLES CAN BE PROGRAMMED HERE
//---------------------------------------------------------------

The sky is the limit ;-)

// if everything was ok...
return new Verdict(0.0, "No problem detected.", Util.NullVisualizer);;

}
protected boolean moreActions(State state);
protected void finishSequence(File recordedSequence);
protected boolean moreSequences();
How has it been used?
MS Office

- Subject application: Microsoft Word 2011
- Robustness test: random action selection
- 18 hour run
- 672 sequences à 200 actions
- 9 crashes
- 6 reproducible crashes
- Effort was approx 1 hour to:
  - System setup (location, configuration files)
  - Augment Action Set (Drag Sources, Drop Targets, Clicks, Double Clicks, Right Clicks, Text to type, ...)
  - Configure cheap oracle (crashes, timeouts, evident error messages)
CTE XL Professional

- CTE XL Professional is a commercial tool for test case design
- Draw a combinatorial tree modeling test relevant aspects
- Generate a set of abstract test cases
- Java application - Eclipse Rich Client Platform (RCP) using Standard Widget Toolkit (SWT)
- Developed and commercialized by Berner&Mattner

- TESTAR was used to test it.
Do experiments with more sophisticated action selection

• What is a “good” test sequence?
  ➔ One that generates lots of Maximum Call Stacks (MCS)
• MCS: root-leaf-path through call tree
• Intuition: the more MCSs a sequence generates, the more aspects of the SUT are tested (McMaster et al.)
• #MCS = number of leaves
• Obtainable through bytecode instrumentation (no source code needed)
Do experiments with more sophisticated action selection

• Select actions in such a way that sequences are formed that generate large amounts of “Maximum Call Stacks” within the system under test (SUT)

• Optimization algorithm used:
  – Ant Colony Optimization
Ant Colony Optimization

- C = component set (here: C = set of feasible actions)
- The likelihood that \( c_i \in C \) is chosen is determined by its pheromone value \( p_{ci} \)
- Generate trails (sequences) by selecting components according to pheromone values \( p_i \)
- Assess fitness of trails (i.e. MSC)
- Reward components \( c_i \) that appear in “good” trails by increasing their pheromones \( p_i \)

(Upon construction of subsequent trails, prefer components with high pheromone values)
Initial experiment results

- Fixed stopping criteria -> 6000 generated sequences
Conclusion

• Implementation works
  – Better than random
  – Solutions improve over time
  – Letting it run until

• Efficiency
  – Sequence generation is expensive ➞ parallelization
  – Frequent restarts of the SUT ➞ might not be suitable for large applications with a significant startup time, e.g. Eclipse
  – ACO good choice?

• Fault sensitivity? ➞ Empirical evaluation needed
Clave Informática

• We met this company at some local test event in Valencia
• Clavei is a private software vendor from Alicante, which
• Specialized for over 26 years in the development Enterprise Resource Planning (ERP) systems for SMEs.
• Main products is ClaveiCon a software solution for SMEs for accounting and financing control
• Current testing is done manually
• Amount of faults found by clients is too high
• Testing needs to be improved
Objectives of the study

• Can our tool be useful for Clave Informatica?
• Can it help them be more effective in finding faults?
• Can this be done in an efficient way, i.e. not taking too much time.
• Restrictions:
  – Clave had no budget to apply the tool themselves
  – So we, the tool developing researchers did that
ClaveiCon

- Written in Visual Basic
- Microsoft SQL Server 2008 database
- Targets the Windows operating systems.

- Store data about product planning, cost, development and manufacturing.
- Provides a realtime view on a company’s processes and enables controlling inventory management, shipping and payment as well as marketing and sales.
Case Study Procedure

1) Planning Phase:
   a) Implementation of Test Environment
   b) Error Definition: Anticipate and identify potential fault patterns.

2) Implementation Phase:
   a) Oracle Implementation: Implement the detection of the errors defined in the previous step.
   b) Action Definition Implementation
   c) Implementation of stopping criteria

3) Testing Phase: run the test

4) Evaluation Phase:
   a) Identify the most severe problems encountered during the run.
   b) The collected information will be used for the refinement of the setup during the next iteration.
Results

• The pre-testing activities:
  – the development or actions, oracles and stopping criteria to setup TESAR takes some initial effort (in our case approximately 26 hours) but will pay off the more often the test is run.

• The manual labor associated to post-testing:
  – inspection of log files,
  – reproduction and comprehension of errors
Are only a tiny fraction of the overall testing time (we spent 1,5 hour of manual intervention during and after tests, compared to over 91 hours of actual unattended testing).

• TESTAR detected 10 previously unknown critical faults, makes for a surprisingly positive result towards believing that TESTAR can be a valuable and resource-efficient supplement for manual testing.
See a video here:
Softeam

• FITTEST partner from France
• Big software company
• SUT selected for evaluating
  TESTAR: Modelio SaaS

• Modelio SaaS:
  – PHP web application
  – For the transparent configuration of distributed environments that run projects created with SOFTEAM’s Modelio Modeling tool
  – Administrators use this application to manage servers and projects that run in virtual environments on different cloud platforms

• Current testing done manually
We measured:

- **Learnability** (questionnaires, work-diaries, performance evaluations)

- **Effectiveness**
  - 17 faults were re injected to evaluate
  - Code coverage

- **Efficiency**
  - Time for set-up, designing and develop
  - Time for running tests
Results

- Some difficulties/resistance/misunderstanding during the learning of programming for powerful oracles
- Testing artifacts produced increased in quality
  - **Red** = Oracle
  - **Green** = Action Set
  - **Blue** = Stopping Criteria

<table>
<thead>
<tr>
<th>Description</th>
<th>Test Suite</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{TS}_{\text{Soft}}$</td>
</tr>
<tr>
<td>Faults discovered</td>
<td>14 + 1</td>
</tr>
<tr>
<td>Did not find IDs</td>
<td>1, 9, 12</td>
</tr>
<tr>
<td>Code coverage</td>
<td>86.63%</td>
</tr>
<tr>
<td>Time spent on development</td>
<td>40h</td>
</tr>
<tr>
<td>Run time</td>
<td>manual</td>
</tr>
<tr>
<td></td>
<td>1h 10m</td>
</tr>
<tr>
<td>Faults diagnosis and report</td>
<td>2h</td>
</tr>
<tr>
<td>Faults reproducible</td>
<td>100%</td>
</tr>
<tr>
<td>Number of test cases</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 3: Comparison between tests

Would you recommend the tool to your colleagues?

Could you persuade your management to invest?
Student course

- Course: 1\textsuperscript{st} year Master
- “Developing Quality Software”
- 34 students working in groups of 2
- Introduction: 10 minutes
- Going through the user manual (10 pages) while doing a small exercise on a calculator: 50 minutes
- After 1 hour the students were setting up tests for MS paint
Future Work

• Still lot that needs to be done!
• Accessibility API works if UI has been programmed “well”
• Research more search-based approaches for action selection
• Research the integration of other test case generation techniques (model-based, combinatorial-based) for action selection
• Design a test spec language that makes it possible to specify actions and oracles without programming Java
• Do more industrial evaluations to compare maintenance costs during regression testing with our tool and capture/replay or visual testing tools
• Extend the tool beyond PC applications (for now we have Mac and Windows plug-ins) to mobile platforms
• Tanja E. J. Vos
• **email:** tvos@pros.upv.es
• **skype:** tanja_vos
• **web:** [http://tavopol.webs.upv.es/](http://tavopol.webs.upv.es/)
• **telephone:** +34 690 917 971