2009 AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE

"HERMES: A TOOL FOR TESTING MOBILE DEVICE APPLICATIONS"

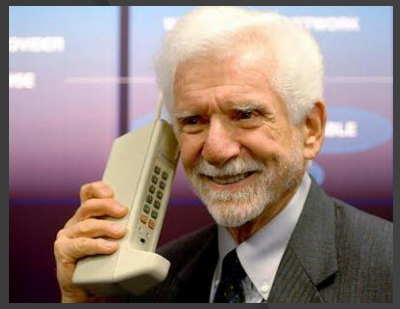
Sakura She, Sasindran Sivapalan, Ian Warren

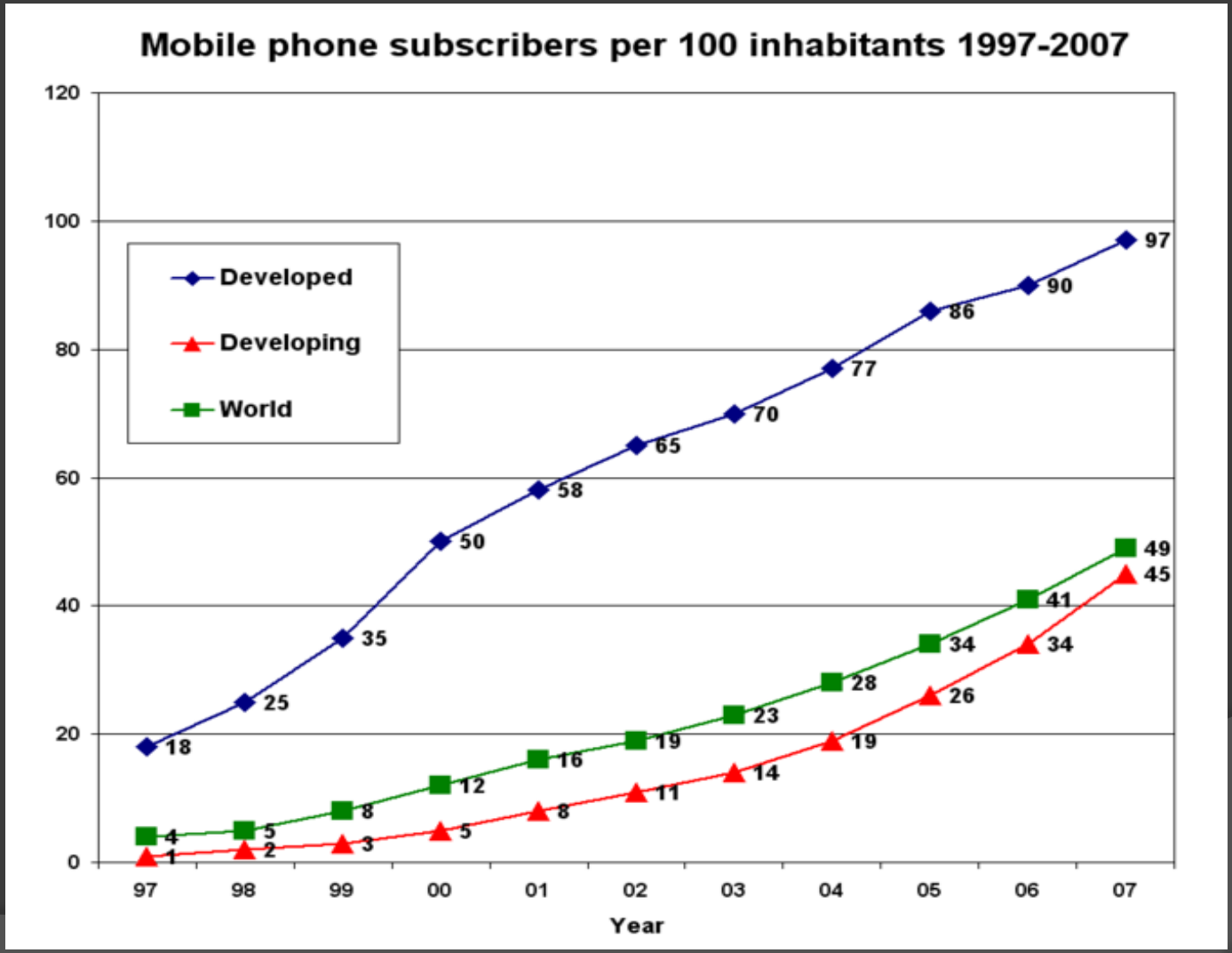
-Ghanashyam Prabhakar

Agenda

- Introduction
- Requirements
- Design
- Evaluation and Results
- Requirement Traceability
- Related Work
- Conclusion, Extensibility and Future Work

Need




Mobile phone subscribers per 100 inhabitants 1997-2007
120
100
80
60
40
20
0
Developed
Developing
World
18
25
35
50
58
65
70
77
86
90
97
4
5
8
12
16
19
23
28
34
41
49
1
2
3
5
8
11
14
19
26
34
45
97
98
99
00
01
02
03
04
05
06
07
Year

Problem

- Heterogeneity in mobile platforms
- Use of Virtual Machines – not so helpful
- Device Emulators – not so helpful

Users


Heir apparent.
Fully functional Mac OS X 10.5 Leopard.
3G compatibility. Stately hard drive capacity.
Ready to ascend the throne as your
all-in-one multimedia device.
The new iPhone. Long live the king.
64GB

<>

Hermes – An Overview

- Supports black box testing of Java applications
- Runs in J2ME environment
- Deploys application on multiple devices
- Executes set of test cases and generates report
- Device independent test cases
- Aesthetics, Functional and Environmental tests

Requirements

1) Automatically deploy applications, execute tests and generate reports
2) Support multi-faceted black-box testing

Facet type	Test concern
Aesthetics	Layout Color Font
Environment	Response time Memory usage Network availability and selection Power consumption
Functional	Content Navigability

Requirements

3) Consume minimal resources on mobile devices
4) Be application independent
5) Be extensible with respect to test case specification and processing
6) Provide benefit at low cost

Design

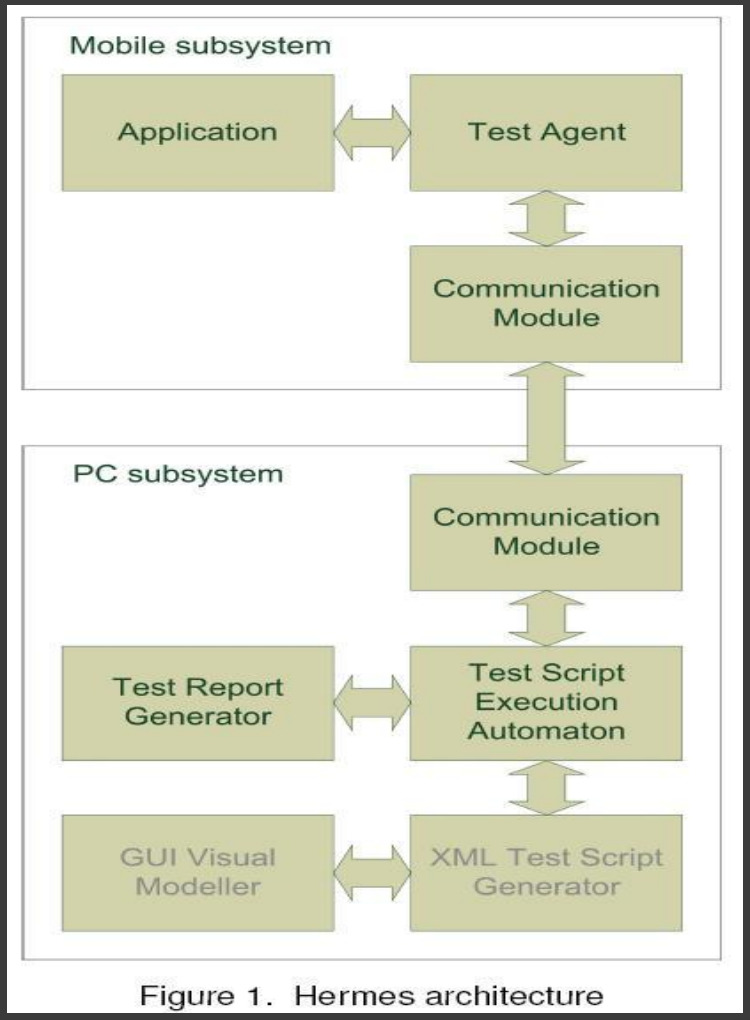


Figure 1. Hermes architecture

Test element	Description
testScript	Exactly one testScript element contains one or more test elements.
test	Describes a test case in terms of remaining elements.
action	A function that is to be simulated on a Midlet application.
componentName	The name of the component to which action relates.
componentType	The type of the component named by componentName, for example Button or TextField.
inputValue	Any value that is to be provided as input by the simulated user.
displayType	The component's display type, such as Form.
expectedOutput	The expected behaviour of the test.
actionDescription	A brief description of the test.

Table 2. XML schema elements for test cases

Communication module - Bluetooth

```
<testScript>
 <test>
  <action>command</action>
  <componentName>OK</componentName>
  <componentType>Command</componentType>
  <displayType>Form</displayType>
  <expectedOutput>Notes</expectedOutput>
  <actionDescription>
   Press the OK button
  </actionDescription>
 </test>
 <test>
  <action>aestheticsLayout</action>
  <componentName>
   Add Food Type
  </componentName>
  <componentType>Form</componentType>
  <displayType>Form</displayType>
  <expectedOutput>
   Component Name=Subject, Alignment=Left;
   Component Name=Deadline, Alignment=Left;
   Component Name=Alarm, Alignment=Left;
   Component Name=Priority, Alignment=Right;
  </expectedOutput>
  <actionDescription>
   Check the alignment of all the components on
   the form
  </actionDescription>
 </test>
 <test>
  <action>getMem</action>
  <componentName>Manager</componentName>
  <componentType>Midlet</componentType>
  <displayType>Midlet</displayType>
  <expectedOutput>
   Less than 200000 bytes
  </expectedOutput>
  <actionDescription>
   Inspect memory usage
  </actionDescription>
 </test>
</testScript>
```

Figure 2. Sample test script

Test Script Preparation

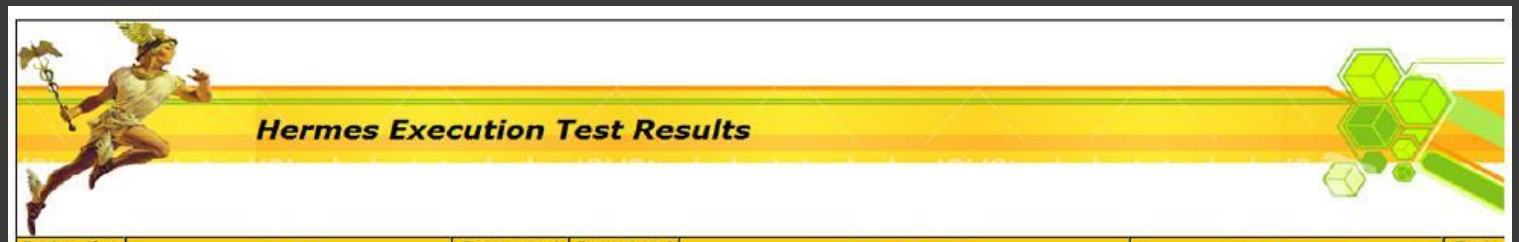
- *GUI Visual Modeller* and *Test Script Generator*
- Generates XML Test Script

Test Script Execution

- *Test Script Execution Automaton, Communication Module, Test Agent*
- Communication Module – Bluetooth
- XML Test Scripts -> Executable Tests
- Sends small number of test requests at a time
- *Test Agent* uses reflection to observe and supply i/p to the AUT(Application Under Test)

Report Generation

- *Test Report Generator*
- Via XSLT, test results to HTML format



Hermes Execution Test Results

Instruction Number	Action	Component Name	Component Type	Expected Output	Actual Output	Test Result
1	*Initiating memory testing*	ToDoManager	Midlet	Any value of the type Long	183160	PASS
2	*Press the OK button*	OK	command	Notes	Notes	PASS
3	*Checking the initial state of the Notes screen*	Notes	List	Name= (no notes), Value= (no notes);	Name= (no notes), Value= (no notes);	PASS
4	*Press the New button*	New	command	Add Note	Add notes	FAIL
5	*Checking the initial state of the Add notes screen*	Add notes	Form	Name= Subject, Value= ;Name= Deadline, Value= Wed Aug 13 01-45-00 UTC 2008;Name= Alarm, Value= ;Name= Priority, Value= High ;	Name= Subject, Value= ;Name= Deadline, Value= Fri Sep 19 15-34-00 GMT+12-00 2008;Name= Alarm, Value= ;Name= Priority, Value= Low ;	FAIL
6	*Checking the alignment of all the components on the screen*	Add notes	Form	Component Name=Subject, Alignment=Left;Component Name=Deadline, Alignment=Left;Component Name=Alarm, Alignment=Left;Component Name=Priority, Alignment=Right;	Component Name=Subject, Alignment=Left;Component Name=Deadline, Alignment=Left; Component Name=Alarm, Alignment=Left;Component Name=Priority, Alignment=Left;	FAIL
7	*Enter nothing into the Subject textfield*	Subject	textfield			PASS
8	*Highlight the Deadline datefield*	Deadline	datefield	Deadline	Deadline	PASS
9	*Enter mon-15-09-2008-17-00-00-000-PM into the Deadline datefield*	Deadline	datefield	mon-15-09-2008-17-00-00-000-PM	Mon Sep 15 17-00-00 GMT+12-00 2008	FAIL
10	*Highlight the Alarm choicegroup*	Alarm	choicegroup	Alarm	Alarm	PASS
11	*Check the alarm checkbox*	Alarm	choicegroup	0	0	PASS
12	*Highlight the Priority choicegroup*	Priority	choicegroup	Priority	Priority	PASS
13	*Choose the Medium priority option*	Priority	choicegroup	1	1	PASS
14	*Press the Save Note button*	Save Note	command	Notes	Notes	PASS
15	*Checking that the Note was not added because it had no Subject specified*	Notes	List	Name= (no notes), Value= (no notes);	Name= , Value= ;	FAIL
16	*Getting the memory usage*	ToDoManager	Midlet	Less than 200000 bytes	42900	79% PASS

Figure 3. Sample report

Evaluation

- Comparative experiment involving use of Hermes vs. Manual Techniques.
  - 2 groups tests 2 applications
  - Faults introduced
  - Time bound, 10 minutes – manual, 25 minutes – Hermes
  - Likert scale questionnaires and feedback
- Comparative experiment of real vs. emulated behavior

Results

92% defects found by Hermes vs. 46% manually - JavaDiet

100% defects found by Hermes vs. 21% manually - ToDoManager

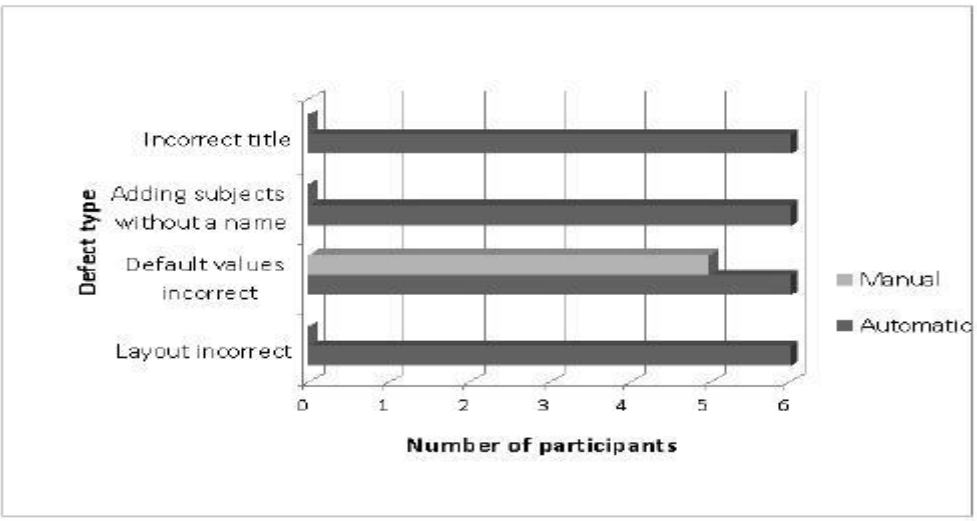


Figure 5. Defect detection for ToDoManager

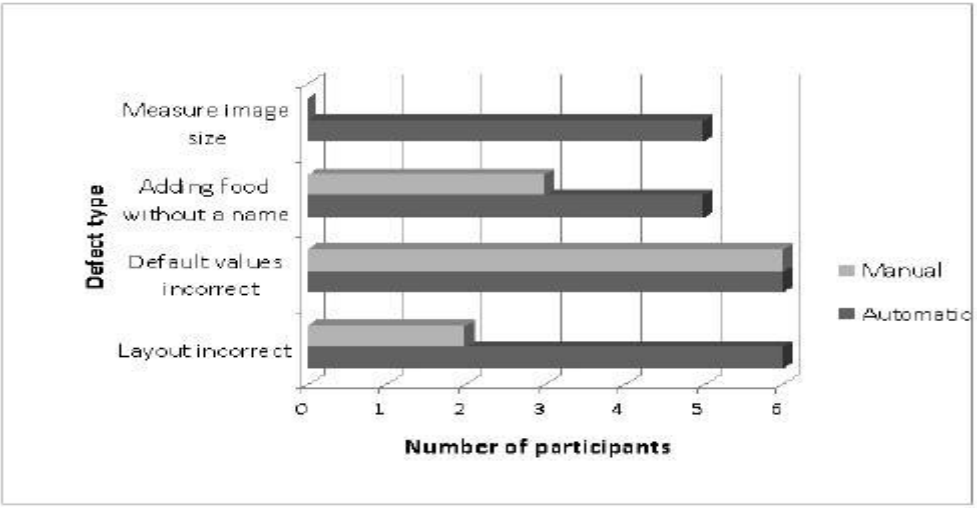


Figure 6. Defect detection for JavaDiet

Requirement Traceability

1) Automatically deploy applications, execute tests and generate reports
   - No automatic deployment

2) Support multi-faceted black-box testing
   - Governed by the capabilities of J2ME API. Ex. GUI layout and component positioning can be inspected but not color

Requirement Traceability

3) Consume minimal resources on mobile devices

   Conserves memory by strategy implemented by the communication module

4) Be application independent

5) Be extensible with respect to test case specification and processing

6) Provide benefit at low cost

Related Work

- Capture / Replay approach – cannot be used until AUT is developed
- Specification based, GUI expressed in XML
- *Test Quest Pro, Digia AppTest, MobileTest* – rely on combination of image comparison, OCR and audio comparison – partial automation

Conclusion, Future Work & Extensibility

- Mainly built to expose the effects of device heterogeneity
- Modular design – hence can be extended
- Missing *Action* issue dealt with by the PC
- Extensive evaluation
- Investigate aggregate reporting
- Developing probes allowing device data to be extracted

Thank you

- for everything else, we test!