Integration and System Testing

Software Testing and Quality Assurance

1

Definitions – Integration Tests

- Integration test data is selected to ensure that the components or sub-systems of a system are working correctly together.
- Test cases will explore different interactions between the components, and make sure the correct results are produced.

2

Definitions – System Tests

- System test data is selected to ensure that the system as a whole is working.
- Test cases will therefore explore the different inputs and combinations of inputs to the system to ensure that the system satisfies its specification

3

Integration Testing – Drivers and Stubs

- Drivers and Stubs are temporary software components
- A test *driver* calls the software under test, passing the test data as inputs.
- In manual testing, where the system interface has not been completed, a test driver is used in its place to provide the interface between the test user and the software under test.

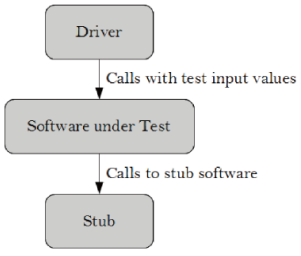
4

Drivers

- Drivers can have varying levels of sophistication.
- It could be hard-coded to run through a fixed series of input values, read data from a prepared file, contain a suitable random number generator etc..

5

Drivers and Stubs - Diagram



6

Stubs

- A *stub* is a temporary or dummy software that is required by the software under test to operate properly.
- This is a throw-away version to allow testing to take place.
- It will provide a fixed or limited set of values to be passed to the software under test.

7

GUI – Celsius to Fahrenheit Example

- This program converts temperature in Celsius to Fahrenheit.
- The driver sends input data, the temperature to be converted, via a windowing system to the user interface under test.
- The core functionality performs the conversion. However, it has not been coded so a stub is written to emulate this.

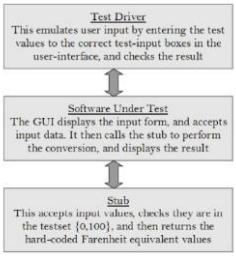
8

GUI – Celsius to Fahrenheit Example

- The user interface calls the stub to do the conversion and display the result.
- The result is picked up by the driver and checked for correctness.
- Only two test case data values are defined (0 and 100 Celsius). The stub can only convert these values.

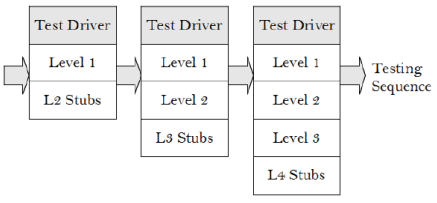
9

GUI – Celsius to Fahrenheit Example



10

Top-Down Integration testing



11

Top-Down Integration testing

- Testing moves from the top layer of the program to the bottom layers.
- Stubs are used to simulate modules being called by the modules under test.

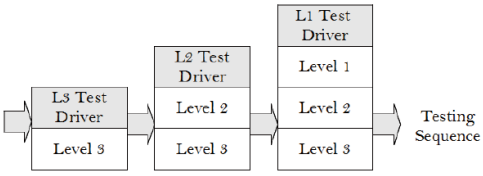
12

Top-Down Integration testing

- Strengths
  - An early outline of the overall program is available so design errors can be found early, giving confidence to the team and customer.
- Weaknesses
  - Proper Stubs are difficult to design.
  - If lower levels are still being created while upper level is complete, sensible changes that could be applied to the upper levels could be ignored.
  - On adding the lower layers, the upper layers will need to be tested again.

13

Bottom-up Integration testing



14

Bottom-up Integration testing

- Strengths
  - Overcomes the disadvantages of top-down testing.
  - Drivers easier to produce then stubs.
  - Developers will have a better understanding of the lower layer modules by the time they reach the top, making them easier to test.
- Weaknesses
  - Harder to imagine the final, working system
  - Important user interaction modules only tested at the end.
  - Drivers need to be created

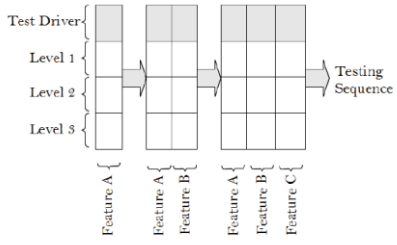
15

Sandwich Integration

- This is a hybrid of Top-down and Bottom-up.
- A target layer is defined somewhere in the middle of the program and testing converges here.
- The top and bottom layer can be tested in parallel and reduces the need for stubs and drivers.
- It is complex to plan and difficult to select the target layer.

16

End-to-end User functionality



17

End-to-end User functionality

- The software is integrated from the bottom up but only small increments of user functionality are added across all the software layers.
- Overcomes problems with adding large amounts of functionality as in the Bottom-up approach.
- The end user can view and test the software as it is being developed, supporting intermediate changes more easily.
- This approach is associated with *Agile Test-driven* development.

18

Integration testing - Considerations

- Number of stubs and drivers needed.
- Location of the critical modules in the system.
- Which layers will be available for testing first.

19

System Testing

- This is testing the system as a whole and is invariably done using black box testing
- System testing can be broken into a number of categories. Not all of these may be carried out on the product or be applicable to it.

20

System Testing - Categories

- **Conformance Testing** Used to verify that the system conforms to a set of published standards. Many standards will have a published suite of conformance tests, or have a selected authority to run these tests. This is particularly important for communications software, as software system must correctly inter-operate with other implementations.
- **Documentation Testing** Used to verify that the documentation (in printed form, online, help, or prompts) is sufficient for the software to be installed and operated efficiently. Typically a full installation is performed, and then the different system functions are executed, exactly as documented. Responses are checked against the documented responses.

21

System Testing - Categories

- **Ergonomic Testing** Used to verify the ease-of-use of the system. This can be either automated (verifying font sizes, information placement on the screen, use of colours, or the speed of progress by users of different experience levels), or manual (based on feedback forms completed by users).
- **Functional Testing** As for Unit Testing, this is used to verify that the system behaves as specified. The interface used for testing is the System Interface which may consistof one or more of: the user interface, the network interface, dedicated hardware interfaces, etc.

22

互操作性测试，用于验证软件可以与其他需要的软件产品交换和共享信息。
通常，在所有对产品上运行测试，以查看需要共享的所有信息是否正确传输。这对通信软件尤其重要。

System Testing - Categories

- **Interoperability Testing** Used to verify that the software can exchange and share information with other required software products. Typically, tests are run on all pairs of products to see that all the information that needs to be shared is transferred correctly. This is particularly important for communications software.
- Note that Conformance Testing is used to make sure that a system conforms to a standard; Interoperability Testing is used to make sure that it actually works with other conformant products. In theory, both are not needed: in practice, it has been found that conformance testing alone is not sufficient.

请注意，一致性测试用于确保系统符合某个标准;互操作性测试用于确保它实际上与其他一致的产品一起工作。在理论上，这两者都不需要:在实践中，已经发现仅仅进行一致性测试是不够的。

23

System Testing - Categories

- **Performance/Load Testing** Used to verify that the performance targets for the software have been met. Some of these tests can be static, but most are dynamic. These tests verify that metrics such as the configuration limits (static), response latency or maximum number of simultaneous users (dynamic) are measured and compared to the specified requirements. Note: just measuring performance is not testing. To test performance, there must be a specification of what is required.

性能/负载测试用于验证软件的性能目标已经达到。其中一些测试可以是静态的，但大多数是动态的。这些测试验证诸如配置限制(静态)、响应延迟或最大并发用户数量(动态)之类的度量，并与指定的需求进行比较。注意:仅仅度量性能不是测试。为了测试性能，必须有一个所需的规范。

24

可移植性测试，用于验证软件可移植到另一个操作环境。在每个新环境中运行来自原始平台的测试选择。
新的平台可能是软件平台(例如同一操作系统的32位和64位版本，或由不同的供应商实现)，
或硬件平台(例如移动电话、平板电脑、笔记本电脑、工作站和服务器)，或不同的环境(数据库、网络等)。

回归测试用于验证对软件的任何更改(bug修复、升级、新特性或对原始应用程序的其他修改)都没有为先前工作的软件引入新的错误。
这是通过重新执行部分或全部的系统测试来执行的。有关在回归测试中使用修复测试的讨论，请参阅第9.6节。
最小化要运行的测试数量是一个活跃的研究领域

System Testing - Categories

- **Portability Testing** Used to verify that the software is portable to another operating environment. A selection of tests from the original platform are run in each new environment. The new platforms may be software platforms (such as 32-bit and 64-bit versions of the same operating system, or implementations by different vendors), or hardware platforms (such as mobile phones, tablets, laptops, workstations, and servers), or different environments (databases, networks, etc.).
- **Regression Testing** Used to verify that any changes to the software (bug fixes, upgrades, new features, or other modifications to the original application) have not introduced new faults to previously working software. This is performed by re-executing some or all of the system tests again. See Section 9.6 for a discussion on using Repair Tests for Regression Testing. Minimising the number of tests to be run is an active research area

25

发布测试用于在发布之前验证产品是否正常运行而不影响应用程序或硬件。通常，系统测试用于测试发布的系统，同时监视其他应用程序和硬件的行为以确保正确运行。

压力测试用于验证失效行为，并在负载超过规定限度时暴露缺陷。用于验证系统是否正常降级。这是一种测试形式，其中的需求可能有些模糊，并且验证软件已经通过测试可能需要解释。通常，软件会承受过多的负载，通常是“输入事件”，测试至少要确保系统不会崩溃。理想情况下，测试将验证是否继续给出有效的响应。

System Testing - Categories

- **Release Testing** Used to verify, prior to distribution, that a product operates properly without interfering with applications or hardware. Typically System Tests are used to exercise the system being released, while the behaviour of other applications and the hardware is monitored for correct operation.
- **Stress Testing** Used to verify failure behaviour and expose defects when the load exceeds the specified limits. Used to verify that the system degrades gracefully or not. This is one form of testing where the requirements may be somewhat vague, and verifying that the software has passed a test may require interpretation. Typically an excessive load is placed on the software, usually in terms of “input events”, and the test is at least to ensure that the system does not crash. Ideally, the testing will verify that valid responses continue to be given.

26

安全性测试用来验证系统的安全特性是否容易受到攻击。这些测试将蓄意破坏安全措施。
例如，冒充其他用户访问受保护的文件、访问网络端口、破坏加密存储或通信、或插入未经授权的软件。

安全测试用于验证系统在安全临界条件下的运行。测试故意在受控条件下引起问题，并验证系统的响应。
对生死攸关的人来说尤其必要。安全关键系统和任务关键系统，如医疗设备软件或航空电子设备。

System Testing - Categories

- **Security Testing** Used to verify that the system security features are not vulnerable to attack. The tests will deliberately attempt to break security measures. Examples would include impersonating another user, accessing protected files, accessing network ports, breaking encrypted storage or communications, or inserting unauthorised software.
- **Safety Testing** Used to verify system operation under safety-critical conditions. The tests deliberately cause problems under controlled conditions and verify the system’s response. Particularly necessary for life-critical, safety-critical, and mission-critical systems, such as medical device software or avionics.

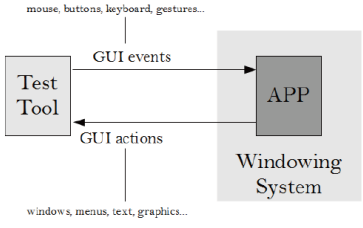
27

System Testing – Test Environment

- A generic test environment model for System Testing is shown in the Figure. The Test Tool provides input (I/P) to the SUT, and receives output (O/P) from the SUT, over the system interface (I/F).
- In some cases the interface will be synchronous, where every input generates an output.
- In other cases the interface will be asynchronous, where an input may create a sequence of outputs over time, or the software may spontaneously generate outputs based on timers or other internal events

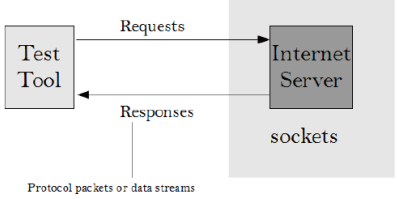
28

System Testing Models - GUI



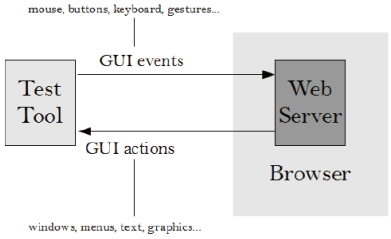
29

System Testing Models - Network



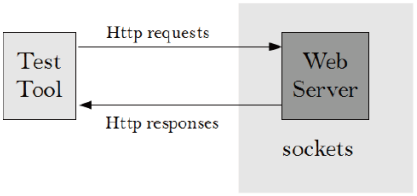
30

System Testing Models – Web based/Browser



31

System Testing Models – Web based/Direct



32

System Testing - GUI Example

- From a testing viewpoint, a GUI consists of multiple windows, each containing various elements. The elements in turn can contain further elements.
- User input takes the form of input events, to which the system responds with actions which cause the results to be displayed to the user.
- A GUI invariably has a defined sequence of windows that can be displayed based on the user input events: for example, displaying a menu, displaying a sub-menu, displaying a new window on top of the existing window (perhaps to set preferences, or save work to a file).

从测试的角度来看，GUI由多个窗口组成，每个窗口包含不同的元素。元素又可以包含更多的元素。
用户输入采用输入事件的形式，系统对其进行响应，并将结果显示给用户。
GUI总是有一个定义好的窗口序列，可以根据用户输入事件显示这些窗口：
例如，显示菜单、显示子菜单、在现有窗口的顶部显示新窗口(可能是为了设置首选项，或者将工作保存到文件中)。

33

System Testing - GUI Example

- System Testing of a GUI can take the following forms:
  - Ensure the interface can be navigated correctly.
  - Ensure the correct elements are on each screen.
  - Verify the correct output actions occur for each input event.

系统测试可以采取以下形式：
-确保界面可以正确导航。
-确保每个屏幕上都有正确的元素。
-验证每个输入事件都发生了正确的输出操作。

34

可以使用状态图覆盖的任何描述的策略。
通常，所有转换都应该针对以下目标：在用户界面中引起更改的每个用户操作都会正确地引起更改

Navigating the interface

- Any of the described strategies for state-diagram coverage can be used.
- Normally *all transitions* should be aimed for: every user action that causes a change in the user interface causes that change correctly

35

如果详细指定了屏幕，可以检查它们以确保它们正确显示。要测试的示例属性
-元素类型：文本框，按钮
-元件布置：抛物线位置，相对位置
-元素值：初始，修改
-元素外观：大小，字体，颜色

Screen Elements

- If screens are specified in detail they can be checked to ensure that they appear correctly. Example attributes to test for
  - Element type: textbox, button
  - Element placement: abolute position, relative position
  - Element values: initial, modified
  - Element appearance: size, font , colour

36

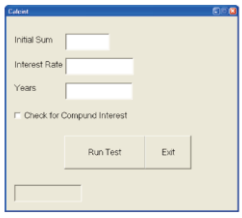
Correct behaviour

- Checked using the methods of black-box testing
- Also consider inputs may be on different windows or menus
- Expected outputs may appear on different windows

37

Simple Interest Calculator GUI

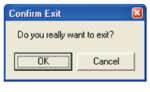
- Considering an example system of a simple interest calculator that has two windows, a Main window



38

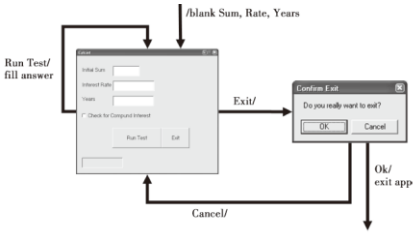
Simple Interest Calculator GUI

- and an Exit window



39

Simple Interest Calculator- GUI State Diagram



40

Simple Interest Calculator- GUI State Diagram

- On entry, the *Calcint* window is displayed and the fields *initial sum*, *interest rate* and *years* are blank.
- If the *Run Test* button is clicked, the *answer* is filled.
- If the *Exit* button is clicked, the *Confirm Exit* window is activated.
- If the *Cancel* button is clicked, the *Confirm Exit* window is deactivated, and focus returns to the *Calcint* window.
- If the *OK* button is pressed, the action *exit app* is taken to terminate the application.

41

Simple Interest Calculator

- To the nearest Euro the interest calculation is given by

$$interest = \begin{cases} initial * (1 + rate) * years & \text{if not compound} \\ initial * (1 + rate)^{years} & \text{if compound} \end{cases}$$

42

System Tests – Navigating the interface

- Use an *every transition* test strategy, where each transition is a test case.
- **Test Cases**
  1. Entry transition
  2. Run Test Transition from Calcint window
  3. Exit transition from Calcint window
  4. Cancel transition from Confirm Exit window
  5. OK transition from Confirm Exit window
- These can be tested within a single test

43

System Tests – Navigating the interface

Test No.	Test Cases Covered	Inputs	Expected Output
1	1, 2, 3, 4, 5	Start application Click Run Test Click Exit Click Cancel Click Exit Click OK	Calcint window active Calcint window active Confirm Exit window active Calcint window active Confirm Exit window active Application exits

44

System Tests – Interface appearance

- The specification states that the text output boxes should be initially blank

45

System Tests – Interface appearance

Test No.	Test Cases Covered	Inputs	Expected Output
2	1, 2, 3, 4, 5	Start application Sum.text() Rate.test() Years.test()	Calcint window active Empty string Empty string Empty string

46

System Tests – Functional Tests

- If Equivalence Partitioning is applied then the following test cases are valid

  1. Initial sum=100
  2. Interest rate=5
  3. Years=10
  4. Compound interest=yes
  5. Compound interest=no

47

System Tests – Functional Tests

Test No.	Test Cases Covered	Inputs	Expected Output
3	1, 2, 3, 4	Initial sum=100 Interest rate=5 Years=10 Compound interest=unchecked Run test	Output=150
4	1, 2, 3, 5	Initial sum=100 Interest rate=5 Years=10 Compound interest=checked Run test	Output=163

48

验收测试(通常称为Alpha和Beta测试)用于验证系统在实际操作环境中是否正确工作。
Alpha测试通常限制在范围内，并在软件开发人员的位置进行。
在Beta测试中，软件通常分发给许多志愿者用户，让他们在自己的环境中运行。
Beta测试对于识别在正常使用过程中发生的故障特别有用——开发团队可能没有考虑到的因素可能在产品发布之前被识别和修复。

Acceptance Testing

- Acceptance Testing (commonly referred to as Alpha and Beta Testing) is used to verify that a system works correctly in a real operational environment.
- Alpha Testing is usually restricted in scope and carried out at the software developer's location.
- In Beta testing, the software is usually distributed for many volunteer users to run in their own environment.
- Beta testing is particularly good for identifying faults that occur during normal use - factors which might not have been considered by the development team may be identified and fixed prior to release of the product.

49

使测试组尽可能的庞大和多样化。
鼓励用户返回报告。
对用户评估保密。

Beta Testing Key Factors

- Making the test group as large and diverse as possible.
- Providing incentives for users to return reports.
- Keeping the user evaluations confidential.

50

客户验收测试由客户使用，以确保他们所订购的系统正常工作。
通常他们会在支付系统之前使用，或者他们可能会用于多个投标解决方案的比较分析。
在某些情况下，测试，或者只是测试的一个子集，被发布给软件开发人员。
在其他情况下，测试是保密的，并且只公布失败测试的细节。

Customer Acceptance Testing

- Customer Acceptance Tests are used by a customer to ensure that the system that they ordered works properly. Typically they would be used prior to payment for the system, or they might be used in a comparative analysis of multiple tendered solutions.
- In some cases the tests, or only a subset of the tests, are released to the software developer.
- In other cases, the tests are kept secret, and details only released of tests that fail.

51