Software Test and Analysis in a Nutshell

資科系
林偉川

1

Course OUTLINE

- Software testing and analysis: process, principles and techniques, Michal Young, 2007
- Midterm ➔30%
- Final exam ➔30%
- Attend the class ➔10%
- 30%
  - Home Work but not everytime

3

Learning objectives

- View the “big picture" of software quality in the context of a software development project and organization:
- Introduce the range of software verification and validation (V&V) activities
- Software testing is one element of a broader topic that is often referred to as verification and validation (V&V).
- Verification –are we building the product correctly? Refers to the set of activities that ensure that software correctly implements a specific function.

4

Learning objectives

- Validation –are we building the correct product?
  Refers to the set of activities that ensure that the software that has been built is traceable to customer requirements
- Provide a rationale for selecting and combining V&V within a software development process.

5

Engineering processes

- Sophisticated tools to amplify developing capabilities but can not remove human error
- Engineering disciplines pair with design and construction activities that check intermediate and final products to identify and remove the defects
- Software engineering is no exception: construction of high quality software requires complementary pairing of design and verification activities

6

Verification and design activities

- Verification and design activities take various forms
  - suited to highly repetitive construction of non-critical items for mass markets
  - suited to highly customized or highly critical products.
- Appropriate verification activities depend on
  - engineering discipline
  - construction process
  - final product
  - quality requirements

7

Peculiarities of software

- Verification grows more difficult with the complexity and variety of the product
- Software has some characteristics that make V&V particularly difficult:
  - Many different quality requirements
  - Evolving (and deteriorating) structure
  - Inherent non-linearity (software development is non-linearity)
  - Uneven distribution of faults

Example

If an elevator can safely carry a load of 1000 kg, it can also safely carry any smaller load;
If a procedure correctly sorts a set of 256 elements, it may fail on a set of 255 or 53 or 12 elements, as well as on 257 or 1023.

8

Impact of new technologies

- Cost of SW verification exceeds half the overall cost of SW development and maintenance
- Advanced development technologies
  - can reduce the frequency of some classes of errors
  - but do not eliminate errors
- New development approaches can introduce new kinds of faults and difficult to reveal and remove

Examples

- With distributed software, the phenomenon of deadlock or race conditions V.S. sequential programs
- New problems due to the use of polymorphism, dynamic binding and private state in object-oriented software.

9

Variety of approaches

- There are no fixed recipes for attacking the problem of verifying a SW product
- Test designers are challenging
  - choose and schedule the right blend of techniques to reach the required level of quality within cost constraints
  - design a specific solution that suits the problem, the requirements, the development environment

10

Five Basic Questions for the Quality Manager

- Computer company to add new online shopping functions to the company Web to allow customers to purchase individual configured products (EC Web)

1. When do verification and validation start? When are they complete?
2. What particular techniques should be applied during development to obtain acceptable quality at acceptable cost?

11

Five Basic Questions for the Quality Manager

3. How can we assess the readiness of a product for release?
4. How can we control the quality of successive releases?
5. How can the development process itself be improved over the course of the current and future projects to improve products and make verification more cost effective?

12

1: When do V&V start? When to complete?

- Test is not a (late) phase of software development
  - Execution of tests is a small part of the verification and validation process to maintain SW quality
- V&V start as soon as we decide to build a software product, or even before
- IT manager to do feasibility study, he should consider not only functionality and development cost, but also the quality and their impact on the overall cost
- V&V last far beyond the product delivery as long as the software is in use, to cope with evolution and adaptations to new conditions

13

Early start: from feasibility study

- At this stage, quality related activities include
  - risk analysis
  - measures needed to assess and control quality at each stage of development
  - assessment of the impact of new features and quality requirements
  - contribution of quality control activities to development cost and schedule
- Add the sales function to the company's website will increase the criticality of system availability and introduce new security issues (SSL, backdoor)

14

Early start: from feasibility study

- Feasibility study that ignored quality could lead to major unanticipated costs and delays which very possible to project failure
- Feasibility study involves some tentative architectural design such as a division of SW structure corresponding to a division of responsibility between a human interface design team and groups responsible for core business software ("business logic") and supporting infrastructure, and a rough build plan breaking the project into a series of incremental deliveries

15

Early start: from feasibility study

- Company design team divides the system into a presentation layer, back-end logic, and infrastructure
- A clean interface between the presentation layer and back-end logic allows a corresponding division between usability testing and verification of correct function
- Considering quality constraints during early breakdown into subsystems allows for a better allocation of quality requirements and facilities both detailed design and testing

16

Early start: from feasibility study

- The initial breakdown of properties given in the feasibility study will be detailed during later design and may result in "cross-quality requirements" among subsystems (to guarantee a given security level), the infrastructure design team may require verification of the absence of some specific security hole (buffer over flow) in other parts of the system (BS779,ISO 9001)
- Acceptance test of the first release will be based on feedback from selected retail stores and the second release will include full acceptance test and reliability measures

17

Early start: from feasibility study

- Even when the project is completed, the SW will continue to evolve and adapt to new conditions or requirements
- V & V activities continue through each small or large change to the system

18

Long lasting: beyond maintenance

- Maintenance activities include
  - analysis of changes and extensions
  - generation of new test suites for the added functionalities
  - re-executions of tests to check for non regression of software functionalities after changes and extensions
  - fault tracking and analysis

19

Long lasting: beyond maintenance

- Making changes to software which is in a known state can, and often does, pose a serious threat to that known state. Every developer knows that even the smallest change can make software useless if the implications of the change were not properly understood or if the change was insufficiently tested during and after implementation.

20

Long lasting: beyond maintenance

- Changes to software are generally unavoidable. Eventually the need arises to amend software, either to correct defects or to modify functionality, and those changes may be required at short notice. It is under these circumstances that the investment in automated regression testing tools can begin to pay back big time.

21

Long lasting: beyond maintenance

- Without automated regression testing tools the emphasis remains on manual testing to re-affirm the 'known' state of software after changes. This may tie up a large number of expensive resources or simply prevent changes from being delivered successfully at short notice. Under such circumstances pressures can lead to the shipping of software which has not been sufficiently re-tested after changes, sometimes with dire consequences.

22

Long lasting: beyond maintenance

- To remain competitive, software developers must be able to implement changes to software quickly and reliably. To support rapid change, testing must be both thorough and quick, leaving little option but to automate the testing process.

23

Long lasting: beyond maintenance

- Successful integration of automated regression testing tools into the development process offers greater levels of confidence that changes introduced into software at short notice will not go without testing nor will they cause unexpected consequences when the software is later shipped to users.
- Automated testing enables thorough testing of software to be conducted both quickly and repeatedly. No longer are expensive resources tied up for long periods of time to repeat tedious manual tests.

24

2: What techniques should be applied?

- No single analysis or test technique can serve all purposes. The primary reasons for combining techniques are:
  - Effectiveness for different classes of faults
    example: conventional testing and testing for race conditions
  - Applicability at different points in a project
    example: inspection for early requirements validation
  - Differences in purpose
    example: statistical testing to measure reliability
  - Tradeoffs in cost and assurance
    example: expensive technique for key properties

25

What techniques should be applied?

- Feasibility study is the first step of a complex development process that should lead to delivery of a satisfactory product through design, verification, and validation activities
- Verification is the process toward the construction of product that satisfies the requirements by checking the quality of intermediate artifacts as well as the ultimate product
- Validation checks the correspondence of the intermediate artifacts and the final product to users' expectation

26

What techniques should be applied?

- The choice of the set of test and analysis techniques depend on quality, cost, scheduling, and resource constraints in development of a particular product
- Business logic subsystem is used a prototype for validating requirement spec. The developer should based on company checklists to identify deviations from design rules for ensuring maintainability, scalability, and correspondence between design and code

27

What techniques should be applied?

- Requirement specs. are written in a structured, semiformal format that are not amenable to automated check, but like other SW artifact and can be inspected by developer
- The company has made a checklist based on their rules for structuring spec. document and on experience with problems in requirements from past system and asking the inspector to confirm each specified property is stated in a form that can be effectively test

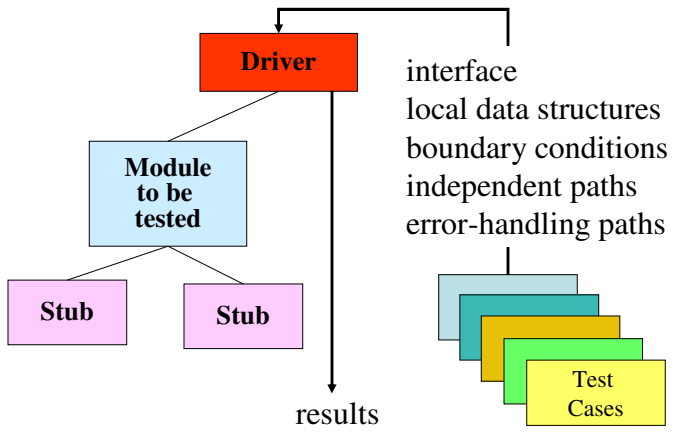
28

What techniques should be applied?

- Component interface specs. are inspected by small groups that include a representative of the "provider" and "consumer" sides of the interface mostly off-line with exchange of notes through a discussion service
- Test stub is additional code needed to execute a unit or a subsystem in isolation
- Test oracles check the results of executing the code and signal discrepancies between actual and expected outputs

29

Unit Test Environment



Note: Redraw from [Pressman 2004]

30

What techniques should be applied?

- Test case are based primarily on interface spec. if less than 90% of all statements are executed by functional tests, this is taken as an indication that either the interface specs are incomplete (missing coverage), or else additional implementation complexity hides behind the interface
- Integration and system test of the some subsystem are modeled as finite state machines, so the quality teams create test suits that exercise program path corresponding to each state transition in the models

31

State machine models

- These model the behaviour of the system in response to external and internal events
- They show the system's responses to stimuli so are often used for modelling real-time systems
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another
- Harel's Statecharts are an integral part of the UML
- A stimulus can trigger a transition from one state to another state

32

Microwave oven operation

- Select the power level(Full or Half)
- Input the cook time
- Press start button and the food is cooked for the given time

➔Rounded rectangle represents the system states

➔Labeled arrow represents stimuli which force the system to change state

33

Microwave oven state description

	State	Description
1	Waiting	The oven is waiting for input. The display shows the current time.
2	Half power	The oven power is set to 300 watts. The display shows alf power'.
3	Full power	The oven power is set to 600 watts. The display shows ull power'.
4	Set time	The cooking time is set to the user's input value. The display shows the cooking time selected and is updated as the time is set.
5	Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
6	Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
7	Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for 5 seconds. Oven light is on. Display shows Cooking complete?w hile buzzer is sounding.

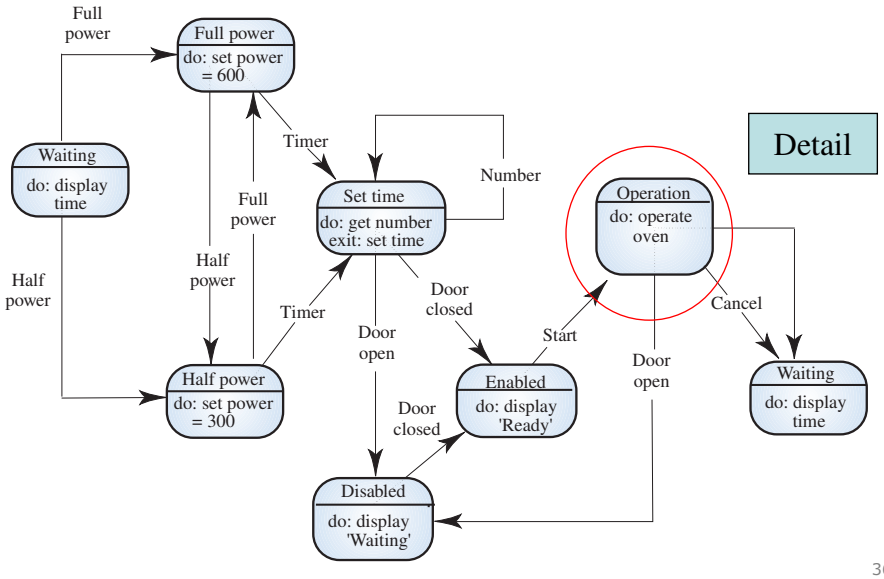
34

Microwave oven stimuli

	Stimulus	Description
1	Half power	The user has pressed the half power button
2	Full power	The user has pressed the full power button
3	Timer	The user has pressed one of the timer buttons
4	Number	The user has pressed a numeric key
5	Door open	The oven door switch is not closed
6	Door closed	The oven door switch is closed
7	Start	The user has pressed the start button
8	Cancel	The user has pressed the cancel button

35

Microwave oven model



36

State machine table for microwave oven

Stimulus / State	1	2	3	4	5	6	7	8
1	2	3						
2		3	4					
3	2		4					
4				4	5	6		
5						6		
6							7	
7					5		1	1

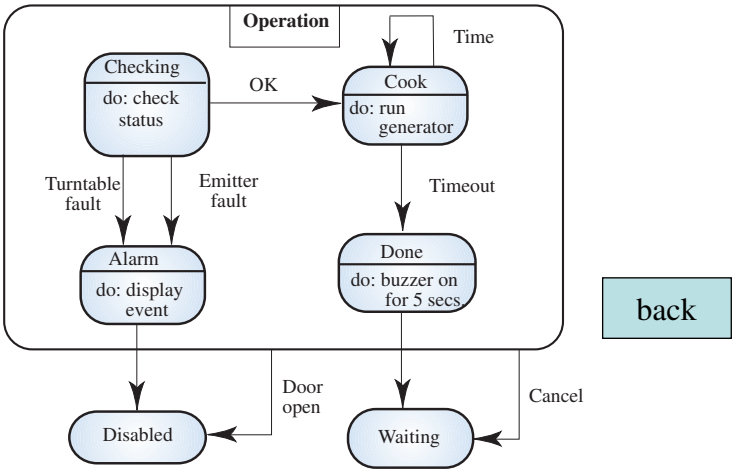
37

Statecharts

- Allow the decomposition of a model into sub-models
- A brief description of the actions is included following the 'do' in each state
- Can be complemented by tables describing the states and the stimuli
- For large system model, a super-state which encapsulates a number of separate states is used for structuring the states structure

38

Microwave oven operation



39

What techniques should be applied?

- The human factors team produces look-and-feel guidelines for the Web purchasing system, which together with a interface design rules can be checked during inspection and test (ergonomics, no web barrier [blind, hearing…])
- The human factors team also produces and executes a usability testing plan

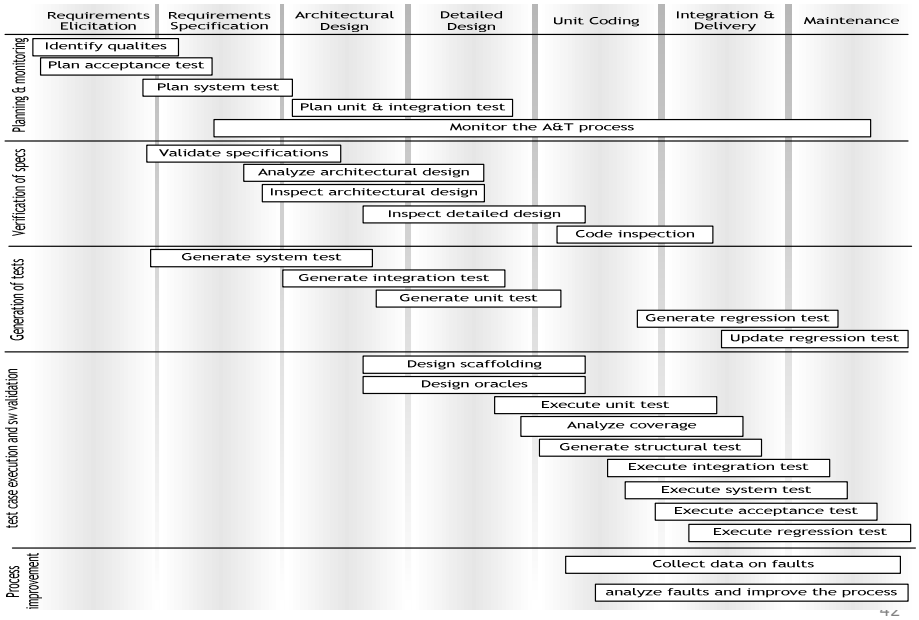
40

What techniques should be applied?

- To construct a coherent plan that can be monitored schedule progress against the plan, the company records faults found during each activity, using this as an indicator of potential trouble spots
- If the number of faults found in a component during design inspections is high, additional dynamic test time will be planned for that component

41

Staging Analysis and Test techniques



42

3: How can we assess the readiness of a product?

- Analysis and Test during development aim at revealing faults
- The truth is that we cannot reveal are remove all faults
- Analysis and Test cannot last indefinitely: we want to know if products meet the quality requirements
- We must specify the required level of dependability and determine when that level has been attained

43

Different measures of dependability

- Availability measures the quality of service (QoS) in terms of running versus down time
- Mean time between failures (MTBF) measures the quality of the service in terms of time between failures ➔ the length of time intervals during which the service is available
- Reliability indicates the fraction of all attempted operations that complete successfully (availability)

44

Different measures of dependability

- Both reliability and availability are important for company's web presentation. The availability goal is set at an average of no more than 30 mins. of down time per month. A reliability goal of less than 1 failure per 1000 user sessions, with a rule to set that certain critical failures must be rare
- 30 one-minute failures in the course of a day would be much worse than a single 30 minutes failure, MTBF is specified as at least one week

45

Different measures of dependability

- From the experience of many previous projects, the company is fruitful to begin measuring reliability. For some application domain, the company has gathered a large amount of historical usage data from which to define an operational profile that are used to generate large, statistically valid sets of randomly generated tests
- Unfortunally, in many cases such an operational profile is not available

46

Example of different dependability measures

Web sales application:

- 50 interactions terminating with a credit card charge.
- The software always operates flawlessly up to the point that a credit card is to be charged, but on half the attempts it charges the wrong amount.

What is the reliability of the system?

- If we count the fraction of individual interactions that are correctly carried out, only one operation in 100 fail: The system is 99% reliable.
- If we count entire sessions, only 50% reliable, since half the sessions result in an improper credit card charge

47

Assessing dependability

- Randomly generated tests following an operational profile
- Alpha test: tests performed by users in a controlled environment, observed by the development organization
- Beta test: tests performed by real users in their own environment, performing actual tasks without interference or close monitoring

48

Assessing dependability

- Company should plan a small alpha test, followed by a longer beta test period in which the SW is made available only in retail outlets
- To accelerate reliability measurement after subsequent revisions of the system, the beta version will be captured many properties of a usage profile

49

4. How can we ensure the quality of successive releases?

- Software test and analysis does not stop at the first release.
- Software products operate for many years, and undergo many changes:
  - They adapt to environment changes
  - They evolve to serve new and changing user requirements.
- Ongoing Quality tasks after delivery include:
  - test and analysis of new and modified code
  - re-execution of system tests
  - extensive record-keeping

50

How can we ensure the quality of successive releases?

- Company should maintain a DB for tracking problems. This DB serves a dual purpose of tracking and prioritizing actual, known program faults and their resolution and managing communication with users who file problem reports
- Even at initial release, the DB usually includes some known faults, because market pressure seldom allow correcting all known faults before product release
- Some problem reported by users are misunderstandings and feature requests, and many distinct reports turn out to be duplicated which are consolidated

51

How can we ensure the quality of successive releases?

- The company supports recording, classification, and automatic re-execution of test cases.
- Company designates major revisions, involving several developers, as "point release", and small revisions as "patch level releases".
- A major point release is likely to repeat a period of beta testing
- Patch level releases are urgent for some customers. Test and analysis for patch level revisions is abbreviated, and automation is particular important for obtaining a reasonable level of assurance with very fast turnaround

52

How can we ensure the quality of successive releases?

- Each point release must undergo complete regression testing before release, but patch level revisions may be released with a subset of regression tests that run unattended overnight
- When fixing one fault maybe introduce a new fault or re-introduce faults that have occurred in the past

53

5. How can the development process itself be improved?

- The same defects are encountered in project after project finishing
- Quality improvement program tracks and classifies faults to identify the human errors that cause weakness in test and analysis that allow errors to remain undetected
- Quality improvement group members are drawn from developers and quality specialists on several projects to produce recommendations that may include modifications to development, test practices, technology support, and management practices

54

Fault analysis and process improvement

1. Define the data to be collected and implements procedures for collecting them
2. Analyze collected data to identify important fault classes
3. Analyze selected fault classes to identify weaknesses in development and quality measures
4. Adjust the quality and development process

55

Quality assurance process

- Process improvement program provides an opportunity to better integrate fault data collection with other practice, including the normal procedure for assigning, tracking, and reviewing development work assignments
- Quality process improvement is the initial data collection is integrated in the same bug tracking system, which is integrated with the revision and configuration control system by company's developers

56

Quality assurance process

- Data from several projects over time are aggregated and classified to identify classes of faults that are important because they occur frequently and may cause particular severe failures, or costly to repair
- These faults are analyzed to understand how they are initially introduced and why they escape detection
- The improvement steps recommended by quality improvement group may include specific analysis or testing steps for earlier fault detection, may also include design rules and modifications to development and even to management practice

57

An example of recommend for process improvement

1. Faults that affect security were given highest priority
2. During Analysis and Test, the identification of several buffer overflow problems that may affect security
3. Faults were due to bad programming practice and were revealed late due to lack of analysis
4. Action plan: Modify programming discipline and environment and add specific entries to inspection checklists

- An important part of each recommended practice is an recommendation for measuring the impact of the change

58

Summary

- The quality process has three different goals:
  - Improving a software product
  - assessing the quality of the software product
  - improving the quality process
- We need to combine several Analysis and Test techniques through the software process
- Analysis and Test depend on organization and application domain.
- Cost-effectiveness depends on the extent to which techniques can be re-applied as the product evolves.
- Planning and monitoring are essential to evaluate and refine the quality process.

59