

## Software Testing and Quality Assurance

### Testing in the Software Process

1

## Big Bang Development

- Big Bang - wait until all the code has been written and then to test the finished product all at once
- It is an attractive option to developers because testing activities do not hold back the progress towards completing the product.

2

## Big Bang Development Drawbacks

- However, it is a very risky strategy as the likelihood that the product will work, or even be close to working can be very low,
- Also, it is particularly dependent on program complexity and program size.
- Additionally, if tests do reveal faults in the program, it is much more difficult to identify their source. It is then necessary to search through the complete program to locate them.

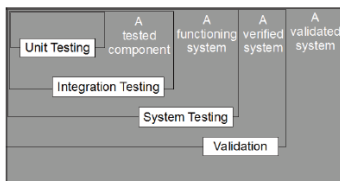
3

## Testing and Development by Stages

- Individual modules, or software features, are tested as they are written.
- This process continues as additional software increments are produced until the product is completed.
- This may have the effect of slowing down the arrival of the final product
- However, it should produce one that has fewer errors and that the development team will have much more confidence in.

4

## Timeline of developing and testing by stages



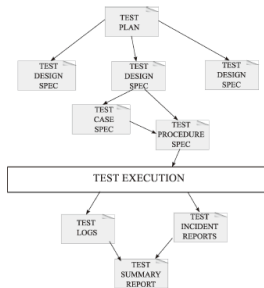
5

## A Test Plan

- A typical Test Plan would include such information as:
  - Items to be tested
  - Tasks to be performed
  - Responsibilities
  - Schedules
  - Required resources

6

## Test Planning – IEEE model



The IEEE standard 892-1998 provides a formal framework within which a plan can be prepared

7

## Software Development Life Cycle

- The Software Development Life Cycle is a structured plan for organizing the development of a software product.
- The need for such planning arose with the growth in size and complexity of software projects.
- By adopting a plan for the development it was intended to create a repeatable and predictable software development process that would automatically improve productivity and quality.

8

## The Waterfall Model

- This model visualizes the software development process as a linear sequence of phases
- It begins with a requirements analysis, followed by the system design, then coding, testing, and ending with system maintenance after the software has been deployed

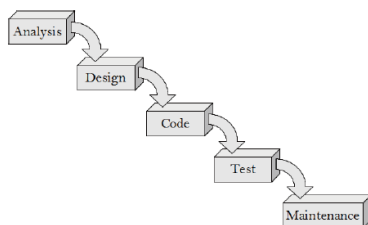
9

## The Waterfall Model

- All the planning is done at the beginning, and once created it is not to be changed.
- There is no overlap between any of the subsequent phases.
- Often anyone's first chance to "see" the program is at the very end once the testing is complete.

10

## The Waterfall Model



11

## Waterfall Model - Strengths

- If time is spent early on making sure that the requirements and design are absolutely correct then this will save much time and effort later.
- There is an emphasis on documentation which keeps all knowledge in a central repository and can be referenced easily by new members joining the team.

12

## Waterfall Model - Weaknesses

- Few visible signs of progress until the end of the project
- It is not flexible to changes
- Time-consuming to produce all the documentation
- Tests are only carried out at the end – this could mean a compromise if time or budgetary constraints exist

13

## Waterfall Model - Weaknesses

- Having to test the program as a whole could result in incomplete testing
- If testing does identify a fault that suggests a redesign it may be ignored because of the trouble involved
- If the customer is unhappy it may incur a long maintenance phase resolving their issues

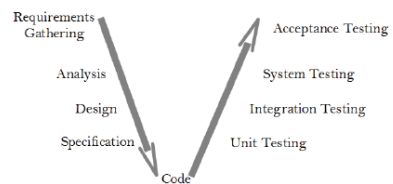
14

## The V-model

- This is an extension of the Waterfall model but in contrast it emphasizes Verification & Validation by marking the relationships between each phase of the life cycle and testing activities.
- Once the code implementation is finished the testing begins.
- This starts with unit testing, and moves up one test level at a time until the acceptance testing phase is completed

15

## The V-Model



16

## V-Model Documentation

- Each document produced is associated with pairs of phases in the model.
- These are the
  - (a) Detailed Design Specifications,
  - (b) the System Design Specifications,
  - (c) the System Requirements Specification,
  - (d) the User Requirements Specification.

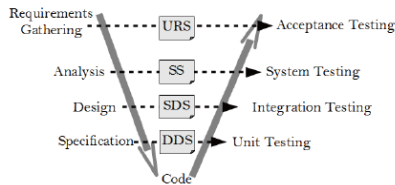
17

## V-Model Documentation

- Requirements Gathering produces the User Requirements Specification (URS), which is both the input to Analysis, and the basis for Acceptance Testing.
- Analysis produces the System Specification (SS) – also known as the Software Requirements Specification (SRS) – which is both the input for Software Design, and the basis for System Testing.
- Design produces the System Design Specification (SDS), which is both the input for the detailed Specification phase, and the basis for Integration Testing.
- The Specification activity produces the Detailed Design Specifications (DDS), which are both used to write the code, and also are the basis for Unit Testing.

18

## V-Model Documentation



19

## V-model Advantages

- It is simple and easy to manage due to the rigidity of the model,
- It encourages Verification and Validation at all phases:
- Each phase has specific deliverables and a review process.
- It gives equal weight to testing alongside development rather than treating it as an afterthought.

20

## V-model Disadvantages

- Its disadvantages are that similarly to the Waterfall model there is no working software produced until late during the life cycle
- It is unsuitable where the requirements are at a moderate to high risk of changing.
- It has been suggested too that it is a poor model for long, complex and object-oriented projects

21

## Agile Development

- Agile methods share with other incremental development methods an emphasis on building releasable software in short time periods.
- However, Agile development differs from the other development models in that its time periods are measured in weeks rather than months and work is performed in a highly collaborative manner

22

## Agile Development

- For effective testing:
  - When the developers “negotiate” the requirements for the upcoming iteration with the customers, the testers must be full participants in those conversations.
  - The testers immediately translate the requirements that are agreed upon in those conversations into test cases.
  - When requirements change, testers are immediately involved because everyone knows that the test cases must be changed accordingly.

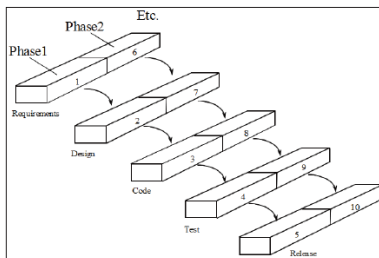
23

## Incremental Development

- The incremental model begins with a simple implementation of a part of the software system. With each increment the product evolves with enhancements being added every time until the final version is reached.
- Testing is an important part of the incremental model and is carried out at the end of each iteration. This means that testing begins earlier in the development process and that there is more of it overall.
- Much of the testing is of the form of regression testing, and much re-use can be made of test cases and test data from earlier increments.

24

## Incremental Development



25

## Incremental Development - Advantages

- A major advantage of the incremental model is that the product is written and tested in smaller pieces, reducing risk and allowing for change to be included easily
- The customers or users are involved from the beginning which means the system is more likely to meet their requirements and they themselves are more committed to the system

26

## Incremental Development - Disadvantages

- It can be difficult to manage because of the lack of documentation in comparison to other models
- The continual change to the software can make it difficult to maintain as it grows in size.

27

## Extreme Programming

- Extreme Programming (XP) is a subset of the philosophy of Agile software development.
- It emphasizes code reviews, continuous integration and automated testing, and very short iterations.
- It favours ongoing design refinement (or *refactoring*), in place of a large initial design phase, keeping the current implementation as simple as possible.
- It favours real-time communication, preferably face-to-face, over writing documents, and working software is seen as the primary measure of progress.

28

## Extreme Programming

- The methodology also emphasizes team work. Managers, customers, and developers are all part of a team dedicated to delivering quality software.
- Programmers are responsible for testing their own work; testers are focused on helping the customer select and write functional tests, and on running these tests regularly

29

## Extreme Programming - Values

- Communication: XP programmers communicate with their customers and fellow programmers
- Simplicity: they keep their design simple and clean
- Feedback: they get feedback by software testing from the start
- Courage: they deliver the system to customers as early as possible and implement changes as suggested, responding with courage to changing requirements

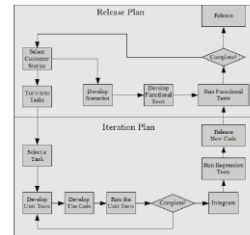
30

## Extreme Programming

- A project begins by identifying a metaphor that describes the system. The metaphor acts as a conceptual framework, identifying key objects and providing insight into their interfaces.
- The first iteration sets the initial skeleton of the project.
- User Stories, in the format of about three sentences of text, are written by the customers. These are the features of the application that the system needs to have and are used to drive the creation of the acceptance tests later on.

31

## Extreme Programming



32

## Extreme Programming

- A Release Plan is created from the User Stories. This plan sets out the overall project.
- Iteration plans are then created for each individual iteration, using development time estimates for each user story.
- The customer specifies scenarios to show that a user story has been correctly implemented. A set of functional (or acceptance) tests is developed based on these.
- The customers are responsible for verifying the correctness of the acceptance tests, and reviewing test scores to decide which failed tests are of highest priority.
- Acceptance tests are also used as regression tests prior to the release of a new version of the software.

33

## Extreme Programming

- Each Iteration Plan is developed in detail just before the iteration begins and not in advance. Iterations are between 1 and 3 weeks in duration.
- User Stories are converted into implementation tasks, recorded on task cards. A programmer takes a task card, writes the unit test cases for the task, implements the code, and tests it.
- When the tests pass, the programmer then integrates the new code, runs regression tests, and releases the code for full functional testing.
- After this, there is a tested, working, software feature ready to demonstrate to the customer.
- Eventually, after all the iterations have been completed the product will be finished.

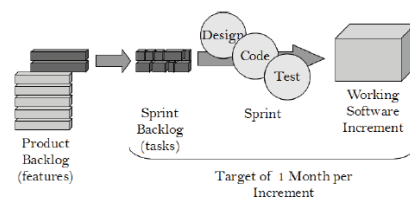
34

## SCRUM – Difference with XP

- Scrum teams work in iterations that are called sprints. These can last a little longer than XP iterations.
- Scrum teams do not allow changes to be introduced during the sprints. XP teams are more flexible with changes within an iteration as long as work has not started on that particular feature already.
- XP implements features in a priority order decided essentially by the customer, while in SCRUM there is more flexibility for additional stakeholders to influence the ordering.
- In XP unit testing and simple design practices are built in, while in SCRUM it is up to the team to organize themselves.

35

## Scrum



36

## Scrum

- SCRUM starts with the Product Backlog which is a prioritized list of all product requirements.
- The backlog items come from: Users, customers, sales, marketing, customer service, engineering, and anyone else that has an interest in the outcome of the project. The Product Backlog is never finalized, it emerges and evolves with the product.
- SCRUM teams take on as much of the product backlog as they think they can turn into an increment of product functionality within a 30-day iteration. This is called a Sprint.
- The team maintains a list of tasks to perform during each Sprint that is called the Sprint Backlog.
- Multiple teams can take on product increments in parallel, all working from the same Product Backlog. A project will have multiple sprints.

37

## Scrum

- Before a sprint starts a meeting is conducted to decide what is going to be developed and delivered in that particular sprint.
- After the completion of the sprint, a meeting is held to collect feedback from the team. This feedback helps in planning and working on the next sprint.
- As the development team starts to work on the next sprint, the testing team carries out functional testing of the features developed in the last sprint.
- This approach gives good results as the testing team works with the developers from the start of the project.

38

## Devops

- DevOps – a combination of **Development & Operations** – is a software development methodology which looks to integrate all the software development functions from development to operations within the same cycle.
- This calls for higher level of coordination within the various stakeholders in the software development process (namely Development, QA & Operations)

39

## Devops

- Requires Continuous Integration combined with Continuous Delivery
- This approach places great emphasis on automation of build, deployment and testing.

40

## Devops

- Build tools help to achieve fast iteration,
- Continuous-Integration (CI) tools to merge code from multiple developers and check for faults.
- Additionally, when the software is in operation, Logging provides traces for identifying and tracking application faults

41

## DevOps and Project Managers

- Complex software architectures and features must be decomposed into small chunks that can be produced and deployed independently.
- Visibility of the configuration and build must be ensured so that everyone is aware of what is deployed, with which versions and dependencies.
- The shift must be made from legacy practices to those that facilitate the integration of development and operations.

42

## DevOps and Testing

- Developers assume responsibility for both the testing and release environment.
- The development team performs test-driven development and CI.
- The use of CI means that the system is constantly being tested

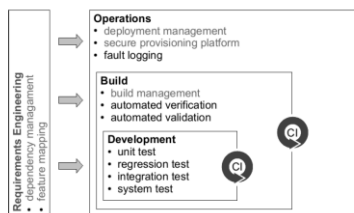
43

## DevOps and Testing

- To accelerate this activity, those responsible for quality assurance must ensure automation of all test cases and full code coverage.
- The intensity of the activities and collaboration in the DevOps environment should therefore result in a noticeable improvement in software quality.

44

## Testing and DevOps



45

## Benefits of DevOps

- Faster Time to market – reduced cycle times and higher deployment rates
- Increased Quality – better availability of the product as it is being created, increased change success rate and fewer failures
- Increased organizational effectiveness – more time spent on value adding activities and greater value being delivered to the customer

46

## Challenges of DevOps

- Complex software architectures and features must be decomposed into small chunks that can be produced and deployed independently.
- Visibility of the configuration and build must be ensured so that everyone is aware of what is deployed, with which versions and dependencies.
- The shift must be made from legacy practices to those that facilitate the integration of development and operations.

47

## Difficulties

- DevOps requires high level of coordination between various functions of the deliverable chain.
- A need to master the various automation and continuous integration tools

48



## Process related quality and standards models - CMM

- The *Capability Maturity Model* was developed initially by the Software Engineering Institute at Carnegie Mellon University in 1991 as a model based on best practices for software development.
- The CMM ranks software development organizations in a hierarchy of five levels, each with a progressively greater capability of producing quality software

49

## CMM Levels

- Level 1 – Initial (also referred to as Chaotic or *Ad Hoc*). Processes are typically undocumented and dynamic. They are driven in an uncontrolled, reactive manner by users or events.
- Level 2 – Repeatable. Processes are repeatable, possibly with consistent results. Process discipline is unlikely to be rigorous, but where it exists it may help to ensure that existing processes are maintained during times of stress.
- Level 3 – Defined. Defined and documented standard processes are established and subject to some degree of improvement over time. These processes are used to establish consistency of process performance across the organization.

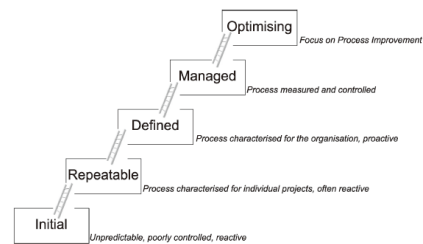
50

## CMM Levels

- Level 4 – Managed. Using process metrics, management effectively controls the process. This includes identifying ways to adjust and adapt the process to particular projects without measurable losses of quality or deviations from specifications.
- Level 5 – Optimizing. The focus is on continually improving process performance through both incremental and innovative technological changes/improvements.

51

## CMM Levels



52

## CMMI

- CMMI defines a number of roles and Software Engineering process areas.
- Testing is mainly associated with the Software Quality Assurance (SQA) and Software Quality Control (SQC) roles in the CMMI model. The main test activities are in the following Software Engineering process areas:
  - CMMI Technical Solution – Unit Testing
  - CMMI Product Integration – Integration Testing
  - CMMI Verification – System Testing
- The test results provide a measure of the process quality, used as a significant input to the Process Assessment and Improvement activity.

53