Software Testing and Quality Assurance

Testing in the Software Process

1

大爆炸——等到所有代码都写好了，然后一次性测试完成的产品。
对于开发人员来说，这是一个有吸引力的选择，因为测试活动不会阻碍完成产品的过程。

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

然而，这是一个非常危险的策略，因为产品成功的可能性很小，甚至接近成功的可能性也很小。
另外，它还特别依赖于程序的复杂性和程序的大小。
此外，如果测试确实揭示了程序中的错误，则很难确定它们的来源。然后有必要搜索整个程序来定位它们。

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

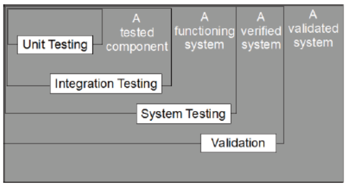
单个模块或软件特性在编写时进行测试。
这个过程会随着额外的软件增量的产生而继续，直到产品完成为止。
这可能会延缓最终产品的到来。
然而，它应该产生一个错误更少，并且开发团队更有信心的。

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



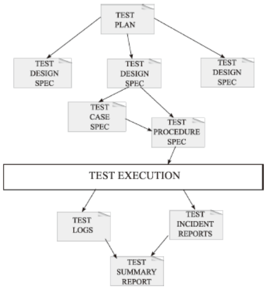
5

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

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.

软件开发生命周期是组织软件产品开发的结构化计划。
对这种规划的需要随着规模的扩大而增加。以及软件项目的复杂性。
通过采用一个开发计划，它的目的是创建一个可重复和可预测的软件开发过程，从而自动提高生产力和质量。

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

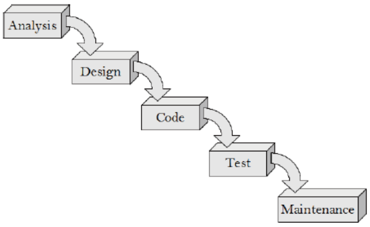
该模型将软件开发过程可视化为一个阶段的线性序列。
它首先进行需求分析，然后进行系统设计，然后编码、测试，最后在软件部署之后进行系统维护。

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.

所有的计划都是在开始的时候完成的，一旦制定了计划就不会改变。
它们在后续阶段之间没有重叠。
通常，任何人第一次 “ 看到 ” 程序是在测试完成之后。

The Waterfall Model



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.

如果将时间花在确保需求和设计绝对正确上，那么这将在以后节省大量时间和精力。
它强调将所有知识保存在一个中央存储库中，并可由加入团队的新成员方便地引用。

直到项目结束之前，很少有明显的进展迹象。
它对变化没有灵活性。
制作所有文档非常耗时。
测试只在最后进行，这意味着如果存在时间或预算限制，就会做出妥协

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

必须测试整个程序可能会导致测试不完整。
如果测试确实发现了一个建议重新设计的错误，那么它可能会因为涉及到的问题而被忽略。
如果客户不满意，解决他们的问题可能需要很长的维护阶段

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

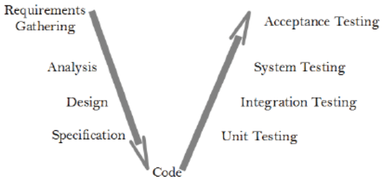
这是瀑布模型的扩展，但与之相反，它通过标记生命周期的每个阶段与测试活动之间的关系来强调验证和确认。
一旦代码实现完成，测试就开始了。这从单元测试开始，每次向上移动一个测试级别，直到验收测试阶段完成

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

生成的每个文档都与模型中的阶段对相关联。
这些是
-(a)详细设计规格;
-(b)系统设计规格;
-(c)系统需求规格,
-(d)用户需求规格。

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

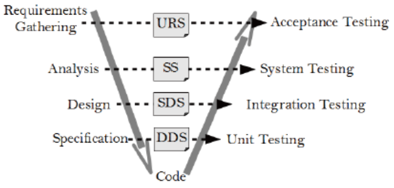
需求收集生成用户需求规范(URS)，它既是分析的输入，又是验收测试的基础。
分析生产系统规范(SS)——也称为软件需求说明书(SRS)——既是软件设计的输入，也是系统测试的基础。
设计生成系统设计规范(SDS)，它是详细规范阶段的输入，也是集成测试的基础。
规范活动生成详细设计规范(DDS)， DDS用于编写代码，也是单元测试的基础。

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 know 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

由于模型的刚性，它简单且易于管理。
它鼓励所有阶段的验证和确认。
每个阶段都有特定的可交付成果和审查过程。
它在开发过程中同样重视测试，而不是把它当作事后的想法。

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

它的缺点是，与瀑布模型类似，在生命周期的后期才会产生可工作的软件。
在需求处于中等到高变化风险的地方是不合适的。
也有人指出，对于长时间、复杂和面向对象的项目来说，它是一个糟糕的模型。

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

敏捷方法与其他增量开发方法一样，强调在短时间内构建可发布的软件。
然而，敏捷开发与其他开发模型的不同之处在于，它的时间周期是以周而不是以月为单位度量的，并且工作是以高度协作的方式执行的。

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

1.增量模型从简单的开始实现了部分软件系统。随着每一个增量，产品都在不断发展，每次都要增加增强功能，直到最终版本的发布。
2.测试是增量模型的重要组成部分，在每次迭代结束时进行。这意味着测试在开发过程中开始得更早，并且总体上有更多的测试。
3.许多测试是回归测试的形式，并且可以对测试用例和来自早期增量的测试数据进行大量重用。

对于有效的测试:
-当开发人员与客户 “ 协商 ” 即将到来的迭代的需求时，测试人员必须是这些对话的完整参与者。
-测试人员立即将这些对话中一致同意的需求转换成测试用例。
-当需求改变时，测试人员马上就会介入，因为每个人都知道测试用例必须随之改变。

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

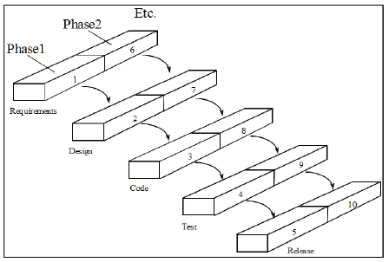
Incremental Development

- 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

增量模型的一个主要优点是，产品是在更小的部分中编写和测试的，这减少了风险，并允许变更可以很容易地包括进来。
客户或用户从一开始就参与其中，这意味着系统更有可能满足他们的需求，他们自己也更忠于系统

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

极限编程(XP)是敏捷软件开发哲学的一个子集。
它强调代码评审、持续集成和自动化测试，以及非常短的迭代。
它支持正在进行的设计优化(或重构)。代替大型的初始设计阶段，保持当前实现尽可能简单。
它更倾向于实时通信，最好是面对面的通信，而不是编写文档，工作软件被视为进步的主要衡量标准。

与其他模型相比，它可能很难管理，因为缺少文档。
随着软件规模的增长，对软件的持续更改可能会使其难以维护。

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

该方法还强调团队合作。经理、客户和开发人员都是致力于交付高质量软件的团队的一部分。
程序员负责测试他们自己的工作;测试人员主要帮助客户选择和编写功能测试，并定期运行这些测试

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

沟通:XP程序员与他们的客户和同事沟通
简单:他们保持他们的设计简单和干净
反馈:他们从一开始就通过软件测试得到反馈
勇气:他们尽可能早地将系统交付给客户，并根据建议实施更改，以勇气响应更改需求

30

项目首先确定一个描述系统的比喻。隐喻充当概念上的模板，识别关键对象并提供对其接口的洞察。
第一次迭代设置项目的初始框架。
用户故事，在格式上约有三句文字，都是由客户写的。这些是系统需要的应用程序的特性，并用于稍后驱动验收测试的创建。

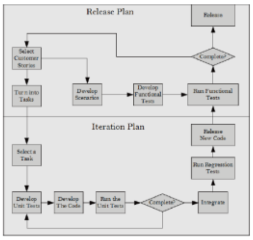
Extreme Programming

- 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

发布计划是根据用户描述创建的。这个计划列出了整个项目。
然后为每个单独的迭代创建迭代计划，使用每个用户故事的开发时间估计客户指定场景来显示用户故事已经正确实现。
在此基础上开发了一组功能(或验收)测试。
客户负责验证验收测试的正确性，并审查测试分数以决定哪些失败的测试是最重要的。
验收测试也用作软件新版本发布之前的回归测试。

Extreme Programmming



32

每个迭代计划都是在迭代开始之前，而不是提前详细开发的。
迭代持续时间在1到3周之间。用户描述被转换为实现任务，并记录在任务卡上。
程序员使用任务卡，为任务编写单元测试用例。
实现代码并对其进行测试当测试通过时，程序员将新代码集成在一起运行回归测试，并发布用于完整功能测试的代码在此之后，将有一个经过测试的、工作的软件特性准备向客户进行演示。
最终，在所有的迭代完成之后，产品就完成了。

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

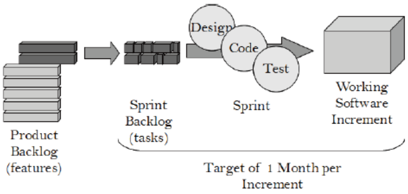
Scrum团队的工作迭代称为sprint。这些比XP迭代持续的时间要长一些。
Scrum团队不允许在sprint中引入变更。
XP团队更灵活地处理在一个周期内的更改，只要这个特定的特性还没有开始工作XP按照优先级顺序实现特性，这基本上是由客户决定的。而在SCRUM中，更多的利益相关者可以更灵活地影响排序。
在XP中，单元测试和简单的设计实践是内置的。而在SCRUM中，则由团队来组织自己。

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


Design
Code
Test
Product Backlog (features)
Sprint Backlog (tasks)
Sprint
Working Software Increment
Target of 1 Month per Increment

36

SCRUM从产品待办事项列表开始，它是所有产品需求的优先级列表。
待定项来自:用户、客户、销售、市场营销、客户服务、工程，以及对项目结果感兴趣的任何人。
产品待办事项列表永远不会最终确定，它会随着产品的出现而发展。
SCRUM团队承担尽可能多的产品待办事项，只要他们认为在30天的迭代中可以转化为产品功能的增量。这叫做冲刺。
团队维护一个在每个Sprint期间要执行的任务列表，该列表称为Sprint Backlog。
多个团队可以并行地进行产品增量，所有团队都从相同的产品待办事项列表中工作。一个项目将有多个sprint。

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.

在sprint开始之前，会召开一次会议，以决定在特定的sprint中要开发和交付什么。
在sprint完成之后，会召开一次会议来收集团队的反馈。这种反馈有助于规划ano进行下一个sprint。
当开发团队开始下一个sprint时，测试团队对上一个sprint中开发的特性进行功能测试。
由于测试团队从项目一开始就与开发人员一起工作，因此这种方法可以提供良好的结果。

38

DevOps—开发和操作的组合—是一种软件开发方法，它将所有的软件开发功能从开发到操作集成在同一个周期内。
这需要在软件开发过程(即开发、QA和运维)中的各种涉众之间进行更高级别的协调。

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

需要与持续交付相结合的持续集成。
这种方法非常强调构建、部署和测试的自动化。

Devops

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

40

构建工具有助于实现快速迭代。
持续集成(CI)工具，用于合并来自多个开发人员的代码并检查错误。
此外，当软件正在操作时，日志记录为识别和跟踪应用程序故障提供了跟踪

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

复杂的软件架构和特性必须分解成小块，这些小块可以独立地生成和部署。
必须确保配置和构建的可见性，以便每个人都知道部署了什么、使用哪个版本和依赖项。
必须将遗留的实践转变为那些促进开发和操作集成的实践。

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

开发人员承担测试和发布环境的责任。
开发团队执行测试驱动开发和CI。
CI的使用意味着系统在不断地进行测试。

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

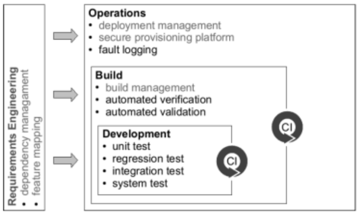
为了加速这些活动负责质量保证必须确保所有测试用例的自动化和全部代码覆盖。
因此，DevOps环境中活动和协作的强度应该会显著提高软件质量。

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


Requirements Engineering
- dependency managament
- feature mapping

Operations
- deployment management
- secure provisioning platform
- fault logging

Build
- build management
- automated verification
- automated validation

Development
- unit test
- regression test
- integration test
- system test

45

更快的上市时间：减少了周期时间和更高的部署率
提高质量：在创建产品时提供更好的可用性，增加更改成功率，减少失败
提高组织的效率：在增加价值的活动上花费更多的时间，并向客户交付更多的价值

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

必须将复杂的软件体系结构和功能分解成小块，这些小块可以独立地生成和部署。
必须确保配置和构建的可见性，以便每个人都知道部署了什么、使用了哪些版本和依赖项。
必须将遗留的实践转变为那些促进开发和操作集成的实践。

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

DevOps要求在交付链的各个功能之间进行高度的协调。
需要掌握各种自动化和持续集成工具

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

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.

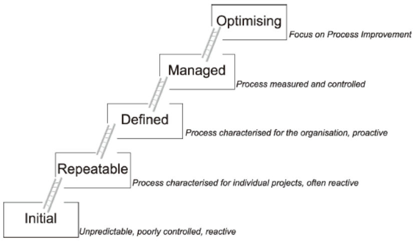
- Level 3 – Defined. Defined and documented standard processes are established and subject to some degree of improvement over time. These processes are used to establish consistency of process performance across the organization.

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.

CMM Levels


Optimising
Focus on Process Improvement
Managed
Process measured and controlled
Defined
Process characterised for the organisation, proactive
Repeatable
Process characterised for individual projects, often reactive
Initial
Unpredictable, poorly controlled, reactive

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.