|  |  |
| --- | --- |
| * Chapter 2 Definitions * Verification: Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled. [page 28] * Validation: Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled. [page 28] * V-model: A framework to describe the software development lifecycle activities from requirements specification to maintenance. The V-model illustrates how testing activities can be integrated into each phase of the software development lifecycle. [page 28] * Integration: The process of combining components or systems into larger assemblies. [page 29] * Test level: A group of test activities that are organized and managed together. A test level is linked to the responsibilities in a project. Examples of test levels are component test, integration test, system test and acceptance test. [page 29] * Off-the-shelf software (commercial off-the shelf software): A software product that is developed for the general market, i.e. for a large number of customers, and that is delivered to many customers in identical format. [page 30] * Performance: The degree to which a system or component accomplishes its designated functions within given constraints regarding processing time and throughput rate. [page 30] * Incremental development model: A development lifecycle where a project is broken into a series of increments, each of which delivers a portion of the functionality in the overall project requirements. The requirements are prioritized and delivered in priority order in the appropriate increment. In some (but not all) versions of this life cycle model, each subproject follows a ‘mini v-model’ with its own design, coding and testing phases. [page 31] * Iterative development model: A development lifecycle where a project is broken into a usually large number of iterations. An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows from iteration to iteration to become the final product. [page 31] * Agile software development: A group of software development methodologies based on iterative incremental development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. [page 32] * Agile manifesto: A statement on the values that underpin agile software development. The values are   + individuals and interactions over process and tools   + Working software over comprehensive documentation   + Customer collaboration over contract negotiation   + Responding to change over following a plan [page 32] * Efficiency testing: The process of testing to determine the efficiency of a software product. [page 35] * Component testing (unit testing, module testing): The testing of individual software components. Note: the ISTQB Glossary also lists program testing as a synonym, which is somewhat confusing, but consistent with the definition given. [page 35] * Stub: A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component. [page 35] * Driver (test driver): A software component or test tool that replaces a component that takes care of the control and/or the calling for a component or system. [page 35] * Robustness testing: Testing to determine the robustness of the software product. [page 35] * Test-driven development: A way of developing software where the test cases are developed, and often automated, before the software is developed to run those test cases. [page 36] * Integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. [page 36] * System testing: The process of testing an integrated system to verify that it meets specified requirements. Note: The ISTQB Glossary derives from Hetzel’s book *The Complete Guide to Software Testing*, and the implied objective of verification might not be adequate (or even appropriate) for all projects when doing system testing. [page 37] * Functional requirement: A requirement that specifies a function that a component or system must perform. [page38] * Non-functional requirement: A requirement that does not relate to functionality, but to attributes such as reliability, efficiency, usability, maintainability and portability. [page 38] * Test environment (test bed): An environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test. [page 38] * Acceptance testing (acceptance, user acceptance testing): Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system. [page 38] * Maintenance: Modification of a software product after delivery to correct defects, to improve performance or other attributes, or to adapt the product to a modified environment. [page 38] * Alpha testing: Simulated or actual operational testing by potential users/customers or an independent test team at the developers’ site, but outside the development organization. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing. [page 39] * Test type: A group of test activities aimed at testing a component or system focused on a specific, test objective, i.e. functional test, usability test, regression test, etc. A test type may take place on one or more test levels or test phases. [page 41] * Beta testing (field testing): Operational testing by potential and/or existing users/customers, at an external site not otherwise involved with the developers, to determine whether or not a component or system satisfies the user/customer needs and fits within the business processes. Beta testing is often employed as a form of external acceptance testing for off-the shelf software in order to acquire feedback from the market. [page 39] * Functional testing: Testing based on an analysis of the specification of the functionality of a component or system. [page 41] * Black-box testing (specification based testing): Testing, either functional or non-functional, without reference to the internal structure of the component or system. [page 41] * Functionality testing: The process of testing to determine the functionality of a software product. [page 41] * Interoperability testing: The process of testing to determine the interoperability of a software product. [page 41] * Security: Attributes of software products that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data. [page 41] * Security testing: Testing to determine the security of the software product. [page 41] * Performance testing: The process of testing to determine the performance of a software product. [page 42] * Maintainability testing: The process of testing to determine the maintainability of a software product. [page 42] * Load testing: A type of performance testing conducted to evaluate the behavior of a component or system with increasing load, e.g. numbers of parallel users and/or numbers of transactions, to determine what load can be handled by the component or system. [page 42] * Stress testing: A type of performance testing conducted to evaluate a system or component at or beyond the limits of its anticipated or specified work loads, or with reduced availability of resources such as access to memory or servers. [page 42] * Usability testing: Testing to determine the extent to which the software product is understood, easy to learn, easy to operate, and attractive to the users under specified conditions. [page 43] * Reliability testing: The process of testing to determine the reliability of a software product. [page 42] * Portability testing: The process of testing to determine the portability of a software product. [page 42] * Functionality: The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions. [page 43] * Reliability: The ability of the software product to perform its required functions under stated conditions for a specified period of time, or for a specified number of operations. [page 43] * Robustness: The degree to which a component of system can function correctly in the presence of invalid inputs or stressful environmental conditions. [page 43] * Usability: The capability of the software to be understood, learned, used and attractive to the user when used under specified conditions. [page 43] * Efficiency: The capability of the software product to provide appropriate performance, relative to the amount of resources used under stated conditions. [page 43] * Maintainability: The ease with which a software product can be modified to correct defects, modified to meet new requirement, modified to make future maintenance easier, or adapted to a changed environment. [page 43] * Portability: The ease with which the software product can be transferred from one hardware or software environment to another. [page 43] * Black-box (specification-based) test design technique: Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure. [page 44] * White-box testing (structural testing, structure-based testing): Testing based on an analysis of the internal structure of the component or system. [page 44] * White-box (structural, structure-based) test design technique: Procedure to derive and/or select test cases, based on an analysis of the internal structure of a component or system. [page 44] * Code coverage: An analysis method that determines which parts of the software have been executed (covered) by the test suite and which parts have not been executed. e.g. statement coverage, decision coverage or condition coverage. [page 44] * Maintenance testing: Testing the changes to an operational system or the impact of a changed environment to an operational system. [page 45] * Impact analysis: The assessment of change to the layers of development documentation, test documentation and components, in order to implement a given change to specified requirements. [page 46] | лава 2 Определения    **Проверка**: потвърждаване чрез изследване и чрез осигуряване на обективни доказателства, че посочените изисквания са изпълнени. [Страница 28]  Валидиране: потвърждаване чрез изследване и чрез осигуряване на обективни доказателства, че изискванията за специфично предназначена употреба или приложение са били изпълнени. [Страница 28]  V-модел: A рамка, за да опише дейностите по развитие на жизнения цикъл на софтуера от спецификацията на изискванията за поддръжка. В V-модел илюстрира как дейности за изпитване може да се интегрира във всяка фаза на жизнения цикъл на разработка на софтуер. [Страница 28]  Интеграция: Процесът на комбиниране на компоненти или системи в по-големи възли. [Страница 29]  Ниво на изпитване: A група на тестови дейности, които се организират и управляват заедно. ниво A тест е свързан с отговорностите в рамките на проекта. Примери за изпитване нива са компонент тест, тест интеграция, система за изпитване и тестване приемане. [Страница 29]  Off-и телевизионни шелф софтуер (комерсиален софтуер на разстояние-рафта): A софтуерен продукт, който е разработен за общия пазар, т.е. за голям брой клиенти, и че не е представено на много клиенти в идентичен формат. [Страница 30]  Изпълнение: Степента, в която една система или компонент изпълнява своите определени функции в рамките на дадените ограничения по отношение на времето за обработка и скорост на пропускателната способност. [Страница 30]  Точкова модел на развитие: жизнения цикъл на развитие, когато даден проект е разделен на поредица от стъпки, всяка от които доставя част от функционалността на общите изисквания по проекта. Изискванията са приоритизирани и доставени в приоритетен ред в съответното увеличение. В някои (но не всички) версии на този модел на жизнения цикъл, всеки подобект следва "мини V-модел" със свой собствен дизайн, кодиране и тестване фази. [Страница 31]  Повтарящ се модел на развитие: жизнения цикъл на развитие, когато даден проект е разделен на по-обикновено голям брой повторения. Един итерация е пълен цикъл развитие, което води до освобождаване (вътрешен или външен) на един изпълним продукт, подмножество на крайния продукт в процес на развитие, което расте от итерация към итерация да се превърне в крайния продукт. [Страница 31]  Agile за разработка на софтуер: A група на методологии за разработка на софтуер въз основа на повтарящ се частичното развитие, където изисквания и решения се развиват чрез сътрудничество между самоорганизиращи крос-функционални екипи. [Страница 32]  Agile манифест: Изявление на основополагащите ценности на гъвкава разработка на софтуер. Стойностите са  частни лица и взаимодействия над процес и инструменти  Работа софтуер през цялостна документация  сътрудничество Customer над договор договаряне  В отговор на промени с течение на следния план [страница 32]  тестване ефективност: Процесът на тестване, за да се определи ефективността на софтуерен продукт. [Страница 35]  тестване Компонент (единица тестване, тестване модул): Тестването на индивидуални софтуерни компоненти. Забележка: ISTQB Речника също изброява тестване програма като синоним, което е малко объркващо, но в съответствие с определението, дадено. [Страница 35]  Мъниче: A скелетна или специална цел прилагане на софтуерен компонент, използвани за разработване или изпитване на компоненти, който призовава или иначе е зависим от него. Тя замества нарича компонент. [Страница 35]  Driver (тест пилот): софтуерен компонент или тест инструмент, който замества компонент, който се грижи за контрола и / или датата на свикване на компонент или система. [Страница 35]  Здравина изпитване: изпитване, за да се определи устойчивостта на софтуерния продукт. [Страница 35]  Тест задвижване развитие: A начин за разработване на софтуер, където се разработват случаите на изпитване, и често автоматизирани, преди софтуерът е разработен, за да стартирате тези тестови случаи. [Страница 36]  Интеграция на тестване: изследване, за да се покаже дефекти в интерфейсите и в взаимодействията между интегрирани компоненти или системи. [Страница 36]  тестване на системата: Процесът на тестване на интегрирана система за да се провери дали тя отговаря на определени изисквания. Забележка: ISTQB Речника произлиза от книгата Hetzel е пълно ръководство за тестване на софтуер, и косвена цел за проверка може да не е достатъчна (или дори подходящо) за всички проекти, когато се прави тестване на системата. [Страница 37]  Функционално изискване: A изискване, което определя функция, която компонент или система, трябва да изпълни. [Page38]  Нефункционалните изисквания: Изискването не се отнася до функционалността, но да атрибути като надеждност, ефективност, използваемост, ремонтопригодност и преносимост. [Страница 38]  Тест среда (тест легло): Среда, съдържаща хардуер, апаратура, симулатори, софтуерни инструменти, и други спомагателни елементи, необходими за провеждане на тест. [Страница 38]  Проверка при приемане (приемане, проверка при приемане на потребителя): Официално изпитване по отношение на потребителските нужди, изисквания, и бизнес процеси, провеждани за да се определи дали има или няма система отговаря на критериите за приемане и да се даде възможност на потребителя, клиенти или друг упълномощен субект да се определи дали не |