Chapter 1 : Software Engineering environments, Software Engineering

Recently reassessed at Level 3 of the Software Engineering Institute's (SEI) Capability Maturity Model(R) (CMM(R)) for software, the SDC specializes in providing EPA programs with a software engineering environment built on the consistent application of proven software development practices.

Requirements validation, Requirements management. Requirement engineering according to Laplante is "a subdiscipline of systems engineering and software engineering that is concerned with determining the goals, functions, and constraints of hardware and software systems. If the feasibility study suggests that the product should be developed, then requirement analysis can begin. If requirement analysis precedes feasibility studies, which may foster outside the box thinking, then feasibility should be determined before requirements are finalized. Requirements analysis topics[edit] Stakeholder identification[edit] See Stakeholder analysis for a discussion of business uses. Stakeholders SH are people or organizations legal entities such as companies, standards bodies which have a valid interest in the system. They may be affected by it either directly or indirectly. A major new emphasis in the s was a focus on the identification of stakeholders. It is increasingly recognized that stakeholders are not limited to the organization employing the analyst. Other stakeholders will include: In a mass-market product organization, product management, marketing and sometimes sales act as surrogate consumers mass-market customers to guide development of the product organizations which regulate aspects of the system financial, safety, and other regulators people or organizations opposed to the system negative stakeholders; see also Misuse case organizations responsible for systems which interface with the system under design those organizations who integrate horizontally with the organization for whom the analyst is designing the system Stakeholder interviews[edit] Stakeholder interviews are a common technique used in requirement analysis. Moreover, the in-person nature of the interviews provides a more relaxed environment where lines of thought may be explored at length. Joint Requirements Development JRD Sessions[edit] Requirements often have cross-functional implications that are unknown to individual stakeholders and often missed or incompletely defined during stakeholder interviews. These cross-functional implications can be elicited by conducting JRD sessions in a controlled environment, facilitated by a trained facilitator, wherein stakeholders participate in discussions to elicit requirements, analyze their details and uncover cross-functional implications. A dedicated scribe and Business Analyst should be present to document the discussion. Utilizing the skills of a trained facilitator to guide the discussion frees the Business Analyst to focus on the requirements definition process. In the former, the sessions elicit requirements that guide design, whereas the latter elicit the specific design features to be implemented in satisfaction of elicited requirements. Contract-style requirement lists[edit] One traditional way of documenting requirements has been contract style requirement lists. In a complex system such requirements lists can run to hundreds of pages. An appropriate metaphor would be an extremely long shopping list. Such lists are very much out of favour in modern analysis; as they have proved spectacularly unsuccessful at achieving their aims; but they are still seen to this day. Provides a checklist of requirements. Provide a contract between the project sponsor s and developers. For a large system can provide a high level description. Weaknesses[edit] Such lists can run to hundreds of pages. It is virtually impossible to read such documents as a whole and have a coherent understanding of the system. Such requirements lists abstract all the requirements and so there is little context This abstraction makes it impossible to see how the requirements fit or work together. This abstraction makes it difficult to prioritize requirements properly; while a list does make it easy to prioritize each individual item, removing one item out of context can render an entire use case or business requirement useless. This abstraction increases the likelihood of misinterpreting the requirements; as more people read them, the number of different interpretations of the envisioned system increase. Necessarily, these documents speak in generality; but the devil, as they say, is in the details. These lists create a false sense of mutual understanding between the stakeholders and developers. These contract style lists give the stakeholders a false sense of security that the developers must achieve certain things. However, due to the nature of these lists, they inevitably miss out crucial requirements which are identified later in the process. Developers can use these discovered requirements to renegotiate the terms and conditions in their favour. These requirements lists are no help in system design, since they do not lend themselves to application. Alternative to requirement lists[edit] As an alternative to the large, pre-defined requirement lists Agile Software Development uses User stories to define a requirement in every day language. Measurable goals[edit] Best practices take the composed list of requirements merely as clues and repeatedly ask "why? Stakeholders and developers can then devise tests to measure what level of each goal has been achieved thus far. Such goals change more slowly than the long list of specific but unmeasured requirements. Once a small set of critical, measured goals has been established, rapid prototyping and short iterative development phases may proceed to deliver actual stakeholder value long before the project is half over. Prototypes[edit] In the mids, prototyping was seen as the best solution to the requirements analysis problem. Prototypes are Mockups of an application. Prototypes help users get an idea of what the system will look like, and make it easier for users to make design decisions without waiting for the system to be built. Major improvements in communication between users and developers were often seen with the introduction of prototypes. Early views of applications led to fewer changes later and hence reduced overall costs considerably. However, over the next decade, while proving a useful technique, prototyping did not solve the requirements problem: Managers, once they see a prototype, may have a hard time understanding that the finished design will not be produced for some time. Prototypes principally help with design decisions and user interface design. However, they can not tell you what the requirements originally were. Designers and end-users can focus too much on user interface design and too little on producing a system that serves the business process. Prototypes can be flat diagrams often referred to as wireframes or working applications using synthesized functionality. Wireframes are made in a variety of graphic design documents, and often remove all color from the design i. This helps to prevent confusion over the final visual look and feel of the application. Use cases[edit] A use case is a technique for documenting the potential requirements of a new system or software change. Each use case provides one or more scenarios that convey how the system should interact with the end-user or another system to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the end-user or domain expert. Use cases are often co-authored by requirements engineers and stakeholders. Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of all of the ways which the intended users could work with the software or system. Use cases do not describe any internal workings of the system, nor do they explain how that system will be implemented. They simply show the steps that a user follows to perform a task. All the ways that users interact with a system can be described in this manner. Software requirements specification[edit] A software requirements specification SRS is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all of the interactions that the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional or supplementary requirements. Non-functional requirements are requirements which impose constraints on the design or implementation such as performance requirements, quality standards, or design constraints. Recommended approaches for the specification of software requirements are described by IEEE This standard describes possible structures, desirable contents, and qualities of a software requirements specification. The following are common categorizations of requirements that relate to technical management: The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, answer the questions posed in the following listing: Where will the system be used? Mission profile or scenario: How will the system accomplish its mission objective? Performance and related parameters: What are the critical system parameters to accomplish the mission? How are the various system components to be used? How effective or efficient must the system be in performing its mission? How long will the system be in use by the user? What environments will the system be expected to operate in an effective manner? Architectural Requirements Architectural requirements explain what has to be done by identifying the necessary system architecture of a system. Structural Requirements Structural requirements explain what has to be done by identifying the necessary structure of a system. Behavioral Requirements Behavioral requirements explain what has to be done by identifying the necessary behavior of a system. Functional Requirements Functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished. Functional requirements analysis will be used as the toplevel functions for functional analysis. Performance Requirements The extent to which a mission or function must be executed; generally measured in terms of quantity, quality, coverage, timeliness or readiness. During requirements analysis, performance how well does it have to be done requirements will be interactively developed across all identified functions based on system life cycle factors; and characterized in terms of the degree of certainty in their estimate, the degree of criticality to system success, and their relationship to other requirements. For example, a requirement for long range or high speed may result in a design requirement for low weight. A pound item that consists of two subsystems might result in weight requirements of 70 pounds and 30 pounds for the two lower-level items. Requirements analysis issues[edit] Stakeholder issues[edit] Steve McConnell, in his book Rapid Development, details a number of ways users can inhibit requirements gathering: Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied. Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client. Attempted solutions[edit] One attempted solution to communications problems has been to employ specialists in business or system analysis. Techniques introduced in the s like prototyping, Unified Modeling Language UML , use cases, and Agile software development are also intended as solutions to problems encountered with previous methods. Also, a new class of application simulation or application definition tools have entered the market. The best of these tools offer: Alain Abran, James W.

Chapter 2 : Software engineering - Wikipedia

Computer software engineers typically work in well-lit offices in comfortable surroundings or in computer laboratories. Most work at least 40 hours a week, but due to the project-oriented nature of the work, they may also have to work evenings and weekends to meet deadlines or solve unexpected technical problems.

Overview[edit] The term "computer-aided software engineering" CASE can refer to the software used for the automated development of systems software, i. The CASE functions include analysis, design, and programming. CASE tools automate methods for designing, documenting, and producing structured computer code in the desired programming language. CASE software supports the software process activities such as requirement engineering, design, program development and testing. Therefore, CASE tools include design editors, data dictionaries, compilers, debuggers, system building tools, etc. CASE also refers to the methods dedicated to an engineering discipline for the development of information system using automated tools. CASE is mainly used for the development of quality software which will perform effectively. History[edit] The ISDOS project at the University of Michigan initiated a great deal of interest in the whole concept of using computer systems to help analysts in the very difficult process of analysing requirements and developing systems. Several papers by Daniel Teichroew fired a whole generation of enthusiasts with the potential of automated systems development. His insights into the power of meta-meta-models was inspiring, particularly to a former student, Dr. Another major thread emerged as a logical extension to the DBMS directory. By extending the range of meta-data held, the attributes of an application could be held within a dictionary and used at runtime. This "active dictionary" became the precursor to the more modern "model driven execution" MDE capability. However, the active dictionary did not provide a graphical representation of any of the meta-data. Under the direction of Albert F. While, at the time of launch, and for several years, the IBM platform did not support networking or a centralized database as did the Convergent Technologies or Burroughs machines, the allure of IBM was strong, and Excelerator came to prominence. CASE tools were at their peak in the early s. The application development tools can be from several sources: Workbenches support only one or a few activities. Environments support a large part of the software process. Workbenches and environments are generally built as collections of tools. Tools can therefore be either stand alone products or components of workbenches and environments. Tools[edit] CASE tools are a class of software that automate many of the activities involved in various life cycle phases. For example, when establishing the functional requirements of a proposed application, prototyping tools can be used to develop graphic models of application screens to assist end users to visualize how an application will look after development. Subsequently, system designers can use automated design tools to transform the prototyped functional requirements into detailed design documents. Programmers can then use automated code generators to convert the design documents into code. Automated tools can be used collectively, as mentioned, or individually. For example, prototyping tools could be used to define application requirements that get passed to design technicians who convert the requirements into detailed designs in a traditional manner using flowcharts and narrative documents, without the assistance of automated design software.

Chapter 3 : software engineering environment | blog.quintoapp.com

software engineering environment A software system that provides support for the development, repair, and enhancement of software, and for the management and control of these activities. A typical system contains a central database and a set of software tools. The central database acts as a.

Oberndorf all of the SEI , and M. Zelkowitz, Computer Standards and Interfaces 15, In an effort to establish interface standards to help the U. This paper presents a report on the lessons learned in the definition and use of this reference model. Use of an environment classification model by Marvin V. Various reference models have been proposed for the classification of features present in an integrated software engineering environment. The results of this mapping and comments on the effectiveness of the models are given. Information Technology Engineering and Measurement Model: The development of gigabit-rate digital transmission methods is being viewed as a precursor to the gradual merging of the cable television, information technology and telephone industries into a heterogeneous set of information technology providers. This concept has been thought of as the national information infrastructure NII or more colloquially as the "information superhighway. Zelkowitz, Journal of Systems and Software, 35, 1, There is considerable interest today in designing open systems that permit tools to be moved freely among various environments on different hardware platforms. In order to develop such systems, terms like open systems, and features for open systems like interoperability, and integration must all be precisely defined. This paper presents a model that is an extension of a service-based reference model for development environments which can be used to formally define these and other related concepts. The Information Technology Engineering and Measurement ITEM Model has been developed to describe the information processing activities of an enterprise, both the automated tasks performed by computer and the manual processes performed by the information technology staff of an organization. The ability to measure the degree of automation within a process, the ability to define the complexity of a process, and the ability to measure technology transition via a concept called technological drift are all metrics that can evolve from this model. In this paper, we address the effects that the environment has on the development process in order to complete a project. In particular, we are interested in how software process steps are actually performed using a typical programming environment. We then introduce a model to measure a software engineering process in order to be able to determine the relative tradeoffs among manual process steps and automated environmental tools. Understanding process complexity is a potential result of this model.

Chapter 4 : Environmental Software Engineer Jobs, Employment | blog.quintoapp.com

Modeling Software Engineering Environment Capabilities by Marvin V. Zelkowitz, Journal of Systems and Software, 35, 1, () There is considerable interest today in designing open systems that permit tools to be moved freely among various environments on different hardware platforms.

Education[edit] Knowledge of computer programming is a prerequisite for becoming a software engineer. These internships can introduce the student to interesting real-world tasks that typical software engineers encounter every day. Similar experience can be gained through military service in software engineering. Software engineer and Software engineering professionalism Legal requirements for the licensing or certification of professional software engineers vary around the world. In the UK, there is no licensing or legal requirement to assume or use the job title Software Engineer. In some parts of the US such as Texas, the use of the term Engineer is regulated by law and reserved only for use by individuals who have a Professional Engineer license. Employment of computer and information technology occupations is projected to grow 13 percent from to , faster than the average for all occupations. These occupations are projected to add about , new jobs. Demand for these workers will stem from greater emphasis on cloud computing, the collection and storage of big data, and information security [32]. Yet, the BLS also says some employment in these occupations are slowing and computer programmers is projected to decline 7 percent from to since computer programming can be done from anywhere in the world, so companies sometimes hire programmers in countries where wages are lower [33]. Due to its relative newness as a field of study, formal education in software engineering is often taught as part of a computer science curriculum, and many software engineers hold computer science degrees and have no engineering background whatsoever. Software engineers work with businesses, government agencies civilian or military , and non-profit organizations. Some software engineers work for themselves as freelancers. Some organizations have specialists to perform each of the tasks in the software development process. Other organizations require software engineers to do many or all of them. In large projects, people may specialize in only one role. In small projects, people may fill several or all roles at the same time. Most software engineers and programmers work 40 hours a week, but about 15 percent of software engineers and 11 percent of programmers worked more than 50 hours a week in Potential injuries in these occupations are possible because like other workers who spend long periods sitting in front of a computer terminal typing at a keyboard, engineers and programmers are susceptible to eyestrain, back discomfort, and hand and wrist problems such as carpal tunnel syndrome. Many IT certification programs are oriented toward specific technologies, and managed by the vendors of these technologies. Broader certification of general software engineering skills is available through various professional societies. The ACM examined the possibility of professional certification of software engineers in the late s, but eventually decided that such certification was inappropriate for the professional industrial practice of software engineering. Software engineers may be eligible for membership of the Institution of Engineering and Technology and so qualify for Chartered Engineer status. This has sparked controversy and a certification war. It has also held the number of P. Eng holders for the profession exceptionally low. The vast majority of working professionals in the field hold a degree in CS, not SE. Given the difficult certification path for holders of non-SE degrees, most never bother to pursue the license. Impact of globalization[edit] The initial impact of outsourcing, and the relatively lower cost of international human resources in developing third world countries led to a massive migration of software development activities from corporations in North America and Europe to India and later: China, Russia, and other developing countries. This had a negative impact on many aspects of the software engineering profession. For example, some students in the developed world avoid education related to software engineering because of the fear of offshore outsourcing importing software products or services from other countries and of being displaced by foreign visa workers. When Asians are leaving work, Europeans are arriving to work. This provides a continuous ability to have human oversight on business-critical processes 24 hours per day, without paying overtime compensation or disrupting a key human resource, sleep patterns. While global outsourcing has several advantages, global - and generally distributed - development can run into serious difficulties resulting from the distance between developers. This is due to the key elements of this type of distance that have been identified as geographical, temporal, cultural and communication that includes the use of different languages and dialects of English in different locations. As with other aspects of software engineering research is ongoing in this and related areas. Related fields[edit] Software engineering is a direct sub-field of engineering and has an overlap with computer science and management science [48]. It is also considered a part of overall systems engineering.

Chapter 5 : Computer Software Engineer Career Working Environment | Career Education Advisor

Designed for all those involved in the subject area, whether undergraduates, postgraduates or software project managers, this is an introductory guide to the questions surrounding Computer Aided Software Engineering (CASE) - discussing its use, the components and future issues of its environments.

Chapter 6 : How to Become a Software Engineer: 13 Steps (with Pictures)

Software Engineering Stack Exchange is a question and answer site for professionals, academics, and students working within the systems development life cycle.

Chapter 7 : Environment naming standards in software development? - Software Engineering Stack Excha

Software Engineering Tools and Environments Outline How did the field evolve? How can tools and environments be classified and compared? What are the main categories?

Chapter 8 : Software Engineering Environments

ality of environments includes support for a single user for programming- in-the-small, coordination and management of multiple users for program- ming-in-the-large, and management of the software development cycle.