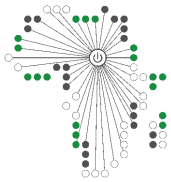


Universidade Virtual Africana

INFORMÁTICA APLICADA: CSI 5302

GARANTIA DE QUALIDADE DE SOFTWARE

José Luís Sambo

Prefácio

A Universidade Virtual Africana (AVU) orgulha-se de participar do aumento do acesso à educação nos países africanos através da produção de materiais de aprendizagem de qualidade. Também estamos orgulhosos de contribuir com o conhecimento global, pois nossos Recursos Educacionais Abertos são acessados principalmente de fora do continente africano.

Este módulo foi desenvolvido como parte de um diploma e programa de graduação em Ciências da Computação Aplicada, em colaboração com 18 instituições parceiras africanas de 16 países. Um total de 156 módulos foram desenvolvidos ou traduzidos para garantir disponibilidade em inglês, francês e português. Esses módulos também foram disponibilizados como recursos de educação aberta (OER) em oer.avu.org.

Em nome da Universidade Virtual Africana e nosso patrono, nossas instituições parceiras, o Banco Africano de Desenvolvimento, convido você a usar este módulo em sua instituição, para sua própria educação, compartilhá-lo o mais amplamente possível e participar ativamente da AVU Comunidades de prática de seu interesse. Estamos empenhados em estar na linha de frente do desenvolvimento e compartilhamento de recursos educacionais abertos.

A Universidade Virtual Africana (UVA) é uma Organização Pan-Africana Intergovernamental criada por carta com o mandato de aumentar significativamente o acesso a educação e treinamento superior de qualidade através do uso inovador de tecnologias de comunicação de informação. Uma Carta, que estabelece a UVA como Organização Intergovernamental, foi assinada até agora por dezenove (19) Governos Africanos - Quênia, Senegal, Mauritânia, Mali, Costa do Marfim, Tanzânia, Moçambique, República Democrática do Congo, Benin, Gana, República da Guiné, Burkina Faso, Níger, Sudão do Sul, Sudão, Gâmbia, Guiné-Bissau, Etiópia e Cabo Verde.

As seguintes instituições participaram do Programa de Informática Aplicada: (1) Université d'Abomey Calavi em Benin; (2) Université de Ougagadougou em Burkina Faso; (3) Université Lumière de Bujumbura no Burundi; (4) Universidade de Douala nos Camarões; (5) Universidade de Nouakchott na Mauritânia; (6) Université Gaston Berger no Senegal; (7) Universidade das Ciências, Técnicas e Tecnologias de Bamako no Mali (8) Instituto de Administração e Administração Pública do Gana; (9) Universidade de Ciência e Tecnologia Kwame Nkrumah em Gana; (10) Universidade Kenyatta no Quênia; (11) Universidade Egerton no Quênia; (12) Universidade de Addis Abeba na Etiópia (13) Universidade do Ruanda; (14) Universidade de Dar es Salaam na Tanzânia; (15) Universite Abdou Moumouni de Niamey no Níger; (16) Université Cheikh Anta Diop no Senegal; (17) Universidade Pedagógica em Moçambique; E (18) A Universidade da Gâmbia na Gâmbia.

Bakary Diallo

O Reitor

Universidade Virtual Africana

Créditos de Produção

Autor

José Luís Sambo

Par revisor(a)

Arlete Maria Vilanculos Ferrão

UVA - Coordenação Académica

Dr. Marilena Cabral

Coordenador Geral Programa de Informática Aplicada

Prof Tim Mwololo Waema

Coordenador do módulo

Jules Degila

Designers Instrucionais

Elizabeth Mbasu

Benta Ochola

Diana Tuel

Equipa Multimédia

Sidney McGregor

Barry Savala

Edwin Kiprono

Kelvin Muriithi

Victor Oluoch Otieno

Michal Abigael Koyier

Mercy Tabi Ojwang

Josiah Mutsogu

Kefa Murimi

Gerisson Mulongo

Direitos de Autor

Apoiado por

Projeto Multinacional II da UVA financiado pelo Banco Africano de Desenvolvimento.

Tabela de conteúdo

Descrição Geral do Curso

Bem-vindo(a) a Garantia de Qualidade de Software (SQA)

O presente curso Garantia de Qualidade de Software, é uma área da engenharia de software que tem como objectivo garantir a qualidade do software através de aplicação de diferente técnicas padronizadas e sistematizadas no processo de desenvolvimento de um software. É também entendida e formada por um grupo de pessoas relacionadas e empregadas através de todo o ciclo de vida de engenharia de software que positivamente influenciam e quantificam a qualidade do software que está sendo entregue.

Segundo Pressman (2001), Garantia de Qualidade de Software não é somente uma actividade associada exclusivamente com actividades de desenvolvimento de software, mas sim atividades que se expandem durante todo o ciclo de vida de desenvolvimento de software isto é, desde o processo até ao produto em conformidade com o padrão ISO 9000, outras técnicas e ferramentas.

Este curso é centrado no desenvolvimento de software com qualidade através do estudo das técnicas e estratégias que garantem essa qualidade(Levantamento de requisitos e análise, verificação e validação, testagem e suas estratégias).

Pré-requisitos

Deverá ter estudado anteriormente:

- Princípios da programação
- Engenharia de Software
- Design e análise Orientado a Objectos

Materiais

Os materiais necessários para completar este curso incluem:

- Computador
- Vídeos on-line
- Páginas da internet
- Livros didáticos
- Objetivos do Curso

Após concluir este curso, o(a) estudante deve ser capaz de:

Ter uma visão geral sobre a gestão de qualidade, abordando diferentes conceitos relacionados;

Conhecer os requisitos de indução e técnicas do desenvolvimento centrado no usuário;

Perceber a principal função de análise de requisitos;

Compreender a natureza de defeitos de software e as diferentes abordagens que levam ao desenvolvimento de software de qualidade;

Equipar os (as) estudantes com o conhecimento, bem como habilidades necessárias para produzir software de qualidade

Conhecer a missão, estratégias e técnicas de testes e;

Saber medir a qualidade de software e identificar as respectivas normas.

Unidades

Unidade 0: Diagnóstico

O propósito desta unidade é verificar o seu nível de conhecimentos de aspectos relacionados com este disciplina.

Unidade 1: Visão geral de SQA

Esta unidade trás um overview sobre conceitos relacionados com a qualidade, qualidade de software e garantia de qualidade de software ou seja a gestão de garantia.

Unidade 2: Requisitos de indução e desenvolvimento centrado no utilizador e Análise de requisitos

Esta unidade irá introduzir o (a) estudante em aspectos sobre como comunicar com o cliente na recolha de requisitos, desenvolvimento de design centrado e como identificar, classificar problemas através de quadros de problemas.

Unidade 3: Verificação e validação

Esta unidade descreve diferentes técnicas de verificação e validação, mostra que o software atende à sua especificação e às necessidades do cliente através de teste.

Unidade 4: Gestão da Qualidade

Este capitulo trata de conteúdos relacionados com as técnicas de medição de qualidade e normas baseadas em modelos de padronização.

Avaliação

Em cada unidade encontram-se incluídos instrumentos de avaliação formativa a fim de verificar o progresso do(a)s estudantes.

No final de cada módulo são apresentados instrumentos de avaliação sumativa, tais como testes e trabalhos finais, que compreendem os conhecimentos construídos e as competências desenvolvidas ao estudar este módulo.

A implementação dos instrumentos de avaliação sumativa fica ao critério da instituição que oferece o curso. A estratégia de avaliação sugerida é a seguinte:

1	Avaliação do tempo de estudo	35%
2	Avaliação do trabalho independente	35%
3	Exame final	30%

Calendarização

Unidade	Temas e Atividades	Estimativa do tempo
0. Revisão de conceitos prévios	• Engenharia de Software • Resolução de Questões.	8 Horas
1. Visão geral de garantia de qualidade de software	• Qualidade de Software; • Elementos da garantia da qualidade; • Tarefas de SQA; • Metas de SQA; • Fiabilidade de Software; • Segurança de Software; • Normas de qualidade ISO 9000; • Plano de SQA.	17 Horas
2. Requisitos de indução, desenvolvimento centrado no utilizador e análise de requisitos.	• Comunicação com clientes sobre os requisitos; • Comunicação com utilizadores e outras partes interessada sobre os requisitos; • Design centrado no usuário; • Avaliar a usabilidade de um produto de software; • Desenvolvimento de software como a resolução de problemas; • Identificação, estruturação e classificação dos problemas através de quadros de Problema.	35 Horas

3. Verificação e validação	• Definição a missão de testes. • Estratégias de teste. • Técnicas de testes de conformidade e; • Desenhos Preliminares de validação através de prototipagem	35 Horas
4. Gestão da Qualidade	• Medição a qualidade de software. • Normas de qualidade de software	25 Horas

Leituras e outros Recursos

As leituras e outros recursos deste curso são:

Unidade 0

Leituras e outros recursos obrigatórios:

Jalote, P. (2012). An integrated approach to software engineering. Springer Science & Business Media.

Roger S Pressman, Engenharia de Software - Uma Abordagem Profissional, ed. Mc Gray Hill, vols., Sétima Edição., 2011.

Leituras e outros recursos opcionais:

Derek Nazareth, "Assembling a Metrics Suite for Rule-Based Systems Development," AMCIS 1999 Proceedings (1999): 23.

Cassiane de Fátima dos Santos Bueno and Gustavo Bueno Campelo, "Qualidade de Software" (n.d.).

Unidade 1

Leituras e outros recursos obrigatórios:

Roger S Pressman, Engenharia de Software - Uma Abordagem Profissional, ed. Mc Gray Hill, vols., Sétima Edição., 2011.

Jalote, P. (2012). An integrated approach to software engineering. Springer Science & Business Media.

Leituras e outros recursos opcionais:

Pankaj Jalote, A Concise Introduction to Software Engineering, ed. Springer, 2008.

Unidade 2

Leituras e outros recursos obrigatórios:

Jalote, P. (2012). An integrated approach to software engineering. Springer Science & Business Media.

Roger S Pressman, Engenharia de Software - Uma Abordagem Profissional, ed. Mc Gray Hill, vols., Sétima Edição., 2011.

Leituras e outros recursos opcionais:

Pankaj Jalote, A Concise Introduction to Software Engineering, ed. Springer, 2008.

Unidade 3

Leituras e outros recursos obrigatórios:

Jalote, P. (2012). An integrated approach to software engineering. Springer Science & Business Media.

Roger S Pressman, Engenharia de Software - Uma Abordagem Profissional, ed. Mc Gray Hill, vols., Sétima Edição., 2011.

Leituras e outros recursos opcionais:

Pankaj Jalote, A Concise Introduction to Software Engineering, ed. Springer, 2008.

Unidade 4

Leituras e outros recursos obrigatórios:

Jalote, P. (2012). An integrated approach to software engineering. Springer Science & Business Media.

Roger S Pressman, Engenharia de Software - Uma Abordagem Profissional, ed. Mc Gray Hill, vols., Sétima Edição., 2011.

Leituras e outros recursos opcionais:

Pankaj Jalote, A Concise Introduction to Software Engineering, ed. Springer, 2008.

Unidade 0. Diagnóstico

Introdução à Unidade

O propósito desta unidade é verificar a compreensão dos conhecimento que possui relacionados com este curso.

Objetivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

Identificar os princípios básicos engenharia de software e sua aplicação;

Indicar as metodologias ou modelos da engenharia de software;

Aplicar diferentes modelos na engenharia de software;

Aplicar os conceitos de modelação de sistemas na resolução de problemas práticos.

Termos-chave

Engenharia de software: É a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software.

Processo de desenvolvimento: é um conjunto de actividades, parcialmente ordenadas, com a finalidade de obter um produto de software.

Avaliação da Unidade

Verifique a sua compreensão!

Avaliação da unidade 0

Instruções

Usando todos recursos bibliográficos fornecidos, responda de uma forma clara às questões que se seguem.

Critérios de Avaliação

Avaliação

1. Quais são as características da Engenharia de Software?
2. Explique a diferença entre verificação e validação?

3. Mencione duas vantagens e desvantagens das seguintes técnicas:

   a. Questionários;

   b. Estudo de Documentação e;

   c. Observação.

4. O que percebes por manutenção de software? Descreve varias categorias de manutenção de sistemas e qual é a categoria que exige mais esforço e porque?

5. O que é UML? Discuta a modelação de sistema usando UML.

6. Explicar como CMM incentiva a melhoria contínua em software, - processo?

Leituras e Outros Recursos

Belgamo, A. & Martins, L.E.G., 2000. Estudo Comparativo sobre as técnicas de Elicitação de Requisitos do Software. In XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná.

Guerra, A.C. & Colombo, R.M.T., 2008. Qualidade de Produto de Software. PBQP/MCT. Disponível em:< http://www. mct. gov. br/index. php/content/view/2867. html# lista>. Acesso em, 13(09), p.2009.

Jalote, P., 2008. A Concise Introduction to Software Engineering Sétima Edição. Springer, ed.,

Nazareth, D., 1999. Assembling a Metrics Suite for Rule-Based Systems Development. AMCIS 1999 Proceedings, p.23.

Pressman, R.S., 2011. Engenharia de Software - Uma Abordagem Profissional Sétima Edição. M. G. Hill, ed.,

Rocha Balthazar, G. da, 1981. Visão Geral da Qualidade de Software. Revista Eletrônica da Faculdade Metodista Granbery-http://re. granbery. edu. br-ISSN, p.0377.

Santos Bueno, C. de F. dos & Campelo, G.B., Qualidade de Software.

Gerard O'Regan, Introduction to Software Quality, Springer, 2014

Jeff Tian, Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement , IEEE Computer Society, 2005

Darrel Ince, An Introduction to Software Quality Assurance and Its Implementation, McGraw-Hill, 1994

Unidade 1. Visão geral da Garantia de Qualidade de Software (SQA).

Introdução à Unidade

Garantia da Qualidade é Conjunto de acções sistematizadas necessárias e suficientes para proveer confiança de que um produto ou serviço irá satisfazer os requisitos definidos da qualidade que, por sua vez, devem refletir as necessidades e as expectativas implícitas e explícitas do cliente. As actividades de garantia da qualidade de software são focadas na prevenção de defeitos e problemas, que podem surgir nos produtos de trabalho. Definição de padrões, metodologias, técnicas e ferramentas de apoio ao desenvolvimento fazem parte deste contexto(Rocha Balthazar, 1981). Assim, Garantia da Qualidade consiste nas funções gerenciais de auditar e relatar. A meta é fornecer à gerência dos dados necessários para que fique informada sobre a qualidade do produto, ganhando assim compreensão e confiança de que a qualidade do produto está satisfazendo suas metas e que para isso um grupo de SQA (Software Quality Assurance) é montado com a missão de ajudar a equipe de software a conseguir um produto final de alta qualidade baseando-se em um conjunto de atividades. (Pressman, 2002)

Objetivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Ter conhecimentos básicos sobre a garantia de qualidade de software;
- Entender a importância de desenvolver os requisitos centrados no usuário;
- Conhecer a técnicas de recolha de requisitos através de comunicação com usuário;
- Desenvolver design centrado no usuário.

Termos-chave

Qualidade de Software: é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos

Garantia da Qualidade: é parte da gestão da qualidade focada em prover confiança de que os requisitos da qualidade serão atendidos

Garantia de Qualidade de Software: é o conjunto de atividades de apoio para fornecer confiança de que os processos estão estabelecidos e estão continuamente melhorados para produzir produtos que atendam as especificações e que sejam adequados para o uso pretendido.

Actividades de Aprendizagem

Actividade 1.1 - Garantia de Qualidade de Software / Software Quality Assurance

Introdução

Esta actividade fornece uma introdução à garantia de qualidade de software. Garantia de qualidade de software (SQA) é a preocupação de cada engenheiro de software para reduzir custos e melhorar o tempo de lançamento do produto no mercado. Ou por outra, é segundo o Pressman (2002) é o conjunto de actividades de apoio para fornecer confiança de que os processos estão estabelecidos e estão continuamente melhorados para produzir produtos que atendam as especificações e que sejam adequados para o uso.

Um Plano de Garantia de Qualidade de Software não é apenas outro nome para um plano de teste, apesar de os planos de teste são incluídas em um plano de SQA. Actividades de SQA são realizadas em cada projecto de software. Uso de métricas é uma parte importante do desenvolvimento de uma estratégia para melhorar a qualidade de ambos os processos de software e produtos de trabalho.

Detalhes da actividade

A garantia de qualidade de software não é uma unidade isolada, para a sua compreenção é necessário perceber as partes que o compõe. Detalhadamente abordamos cada elemento que faz parte do Software Quality Assurance(SQA) ou Garantia de Qualidade de Software.

Qualidade

A palavra qualidade segundo o dicionário pode ser discutida como propriedade, atributo ou condição das coisas ou das pessoas capaz de distingui-las das outras e de lhes determinar a natureza.

Como um atributo de um item, a qualidade se refere a coisas que podem ser medidas, ou seja, comparadas com padrões conhecidos, tais como, tamanho, cor, propriedades elétricas, maleabilidade, etc. Entretanto, é mais difícil categorizar a qualidade de software, pois é uma entidade lógica, do que em objetos físicos.

Controle de Qualidade

Pela definição da ISO, controle de qualidade é a atividade e técnica operacional que é utilizada para satisfazer os requisitos de qualidade.

O controle de qualidade é feito através de uma série de inspeções, revisões e testes, usados através do ciclo de desenvolvimento para garantir que cada trabalho produzido está de acordo com sua especificação. Portanto, o controle de qualidade é parte do processo de desenvolvimento e, como é um processo de feedback, ele é essencial para minimizar os defeitos produzidos.

Qualidade de software

Conjunto de características a serem satisfeitas em um determinado grau de modo que o software satisfaça às necessidades de seus usuários. Preesman (2009) define qualidade de software como um processo de software eficaz aplicadas de um modo que cria um produto útil que fornece um valor mensurável para aqueles que produzem e aqueles que o utilizam. Ela enfatiza três pontos essenciais:

- Processo de software eficaz, produto útil e adicionando valor tanto para o produtor e utilizador.

O processo de software eficaz estabelece a infraestrutura que suporta qualquer esforço para a construção de um produto de software de alta qualidade;

O produto útil entrega o conteúdo, funções e recursos que o usuário final de desejos, mas é importante que ele oferece estes ativos de uma forma fiável, livre de erros e;

Adicionando valor tanto para o produtor e o utilizador de um produto de software de alta qualidade proporciona benefícios para a organização e o usuário final.

Garantia de Qualidade de software

A garantia de qualidade de software não é algo com o qual se começa a pensar depois que o código é gerado. A Garantia de Qualidade de Software ou SQA (Software Quality Assurance) é uma actividade ambrela aplicada ao longo de todo o processo de engenharia de software (Nazareth, 1999).

Custo de Qualidade

Como seria utópico alcançar a perfeição em um sistema de computação, então o mais importante passa a ser definir qual nível de checagem (de qualidade) seria suficiente para o sistema em questão e os custos associados.

O custo de qualidade inclui todos os gastos financeiros relacionados às atividades de qualidade, os quais podem ser divididos em: custos de prevenção, custos de avaliação e custos de falhas.

Os custos de prevenção incluem:

Planificação da qualidade;

Revisões técnicas formais;

Teste de equipamentos;

Treinamento.

Os custos de avaliação incluem:

Inspeções dos processos e relações entre eles;

Manutenção dos equipamentos;

Testes.

Os custos de falhas poderiam desaparecer se nenhum defeito ocorresse antes da entrega do produto para o cliente. Os custos de falhas podem ser divididos em: custos de falhas internas e custos de falhas externas.

Os custos de falhas internas incluem:

Retrabalho;

Conserto de bugs;

Análise de falhas.

Os custos de falhas externas incluem:

Resolução de queixas;

Troca/devolução do produto;

Suporte em help online;

Trabalhos de segurança.

Os custos relacionados a encontrar e consertar um defeito aumentam drasticamente quando vamos da prevenção de falhas internas para a prevenção de falhas externas.(Santos Bueno & Campelo, n.d.)

A figura abaixo representa o custo de qualidade:

Atributos de Qualidade de Software

Segundo o Nazareth (1999), a qualidade de software não é uma ideia tão simples. É mais fácil descrevê-la através de um conjunto de atributos ou factores requeridos que variam de acordo com as diferentes aplicações e os clientes que as solicitam.

Existem várias formas de se classificar os factores de qualidade. Uma delas é classificá-los como factores externos e factores internos.

Factores externos são aqueles cuja presença ou falta num produto de software pode ser detectada pelos usuários do produto (velocidade, facilidade de uso).

Factores internos são aqueles que são perceptíveis apenas por profissionais de computação (modularidade).

Apesar de apenas os factores externos terem importância no final, a chave para assegurar que eles são satisfeitos são os factores internos, ou seja, as técnicas internas são um meio para atingir qualidade de software externa.

Alguns atributos externos são:

- Correctude;
- Robustez;
- Extensibilidade;
- Reusabilidade;
- Compatibilidade;
- Eficiência;
- Portabilidade;
- Verificabilidade;
- Integridade e facilidade de uso.

Outra maneira de se classificar os atributos de qualidade é dividí-los em atributos funcionais e atributos não funcionais.

Os atributos funcionais tipicamente se aplicam a pedaços do software, módulos do sistema como um todo e estão mais relacionados com o quê deve ser feito.

Os atributos não funcionais podem se aplicar a qualquer produto do processo de desenvolvimento: especificações, código, manuais, etc., e estão mais relacionados com o quão bem deve ser feito.

Conclusão

Neste actividade foram discutidos conceito relacionado com a Garantia de Qualidade de Software SQA, onde debruçou-se sobre os conceitos de: qualidade, qualidade de software, controle de qualidade, custo de qualidade e atributos de qualidade e garantia de qualidade de software.

Avaliação

1. Faça uma abordagem explanatória sobre Garantia de Qualidade de software?
2. O que entende por:

   Qualidade;

   Qualidade de software.
3. Explique por palavra suas o que é controlo de qualidade?

Actividade 1.2 - Elementos de garantia de software

Introdução

Esta actividade tem como objectivo discutir e analisar os elementos de garantia de qualidade, as funções de garantia de qualidade de software, as metas e a formalidade de SQA.

Detalhes da atividade

A presente actividade visa a discutir os principais elementos de garantia da qualidade de software que é composta por uma série de tarefas associadas a dois elementos distintos — os engenheiros de software que realizam o trabalho técnico e um grupo de SQA que tem a responsabilidade pela planificação, supervisão, manutenção de registros, análise e relatórios referentes à garantia da qualidade.

Elementos de garantia de qualidade

Os elementos de garantia de qualidade de software podem ser sistematizados da seguinte maneira:

Padrões

O IEEE, a ISO e outras organizações de padronizações produziram uma ampla gama de padrões para engenharia de software e os respectivos documentos.

Esses padrões podem ser adotados para assegurar que as normas sejam adotadas e seguido;

Monitorar continuamente os processos de garantia de qualidade e atualizações em conformidade com os requisitos de qualificação, consulta com as partes interessadas e tomar medidas em feedback.

Revisões e auditorias

As revisões técnicas são uma actividade de controle de qualidade realizada por engenheiros de software para engenheiros de software com intuito de o de revelar erros. Auditorias são um tipo de revisão efetuado pelo pessoal de SQA com o intuito de assegurar-se de que as diretrizes de qualidade estejam sendo seguidas no trabalho de engenharia de software.

Testes

Os testes de software são uma função de controle de qualidade com um objetivo principal — descobrir erros. O papel da SQA é garantir que os testes sejam planificados apropriadamente e conduzidos eficientemente de modo que se tenha a maior probabilidade possível de alcançar seu objetivo primário.

Coleta de análise de erro/defeito

Recolher e analisar dados de erro e defeito para entender melhor como os erros são introduzidos e pode ser eliminada.

Gerenciamento de mudanças:

As mudanças são um dos aspectos mais negativos de qualquer projeto de software. Se não forem administradas apropiadamente, podem gerar confusão, e confusão quase sempre leva a uma qualidade inadequada. A SQA garante que práticas adequadas de gerenciamento de mudanças tenham sido instituídas.

Educação.

Toda organização de software quer melhorar suas práticas de engenharia de software. Um factor fundamental para o aperfeiçoamento é a educação dos engenheiros de software, seus gerentes e outros interessados. A organização de SQA assume a liderança no processo de aperfeiçoamento do software e é um proponente fundamental e patrocinador de programas educacionais.

Gerência dos fornecedores.

O papel do grupo de SQA é garantir software de alta qualidade por meio da sugestão de práticas específicas de garantia da qualidade que o fornecedor deve (sempre que possível) seguir, e incorporar exigências de qualidade como parte de qualquer contrato com um fornecedor externo.

Administração e segurança

Garantir que o software não tenha sido alterado internamente, sem autorização. A SQA garante o emprego de processos e tecnologias apropriadas para ter a segurança de software desejada.

Proteção

Responsável pela avaliação de impacto de falhas de software e iniciar medidas para reduzir o risco.

Gestão de Risco

Embora a análise e a redução de riscos seja preocupação dos engenheiros de software, o grupo de SQA garante que as atividades de gestão de riscos sejam conduzidas apropriadamente e que os planos de contingência relacionados a riscos tenham sido estabelecidos. A importância da gestão de risco advém das inerentes incertezas que muitos

projetos enfrentam. Estes poderiam ser através das falhas nas definições dos requisitos, dificuldades na previsão do tempo e de recursos necessários para o desenvolvimento do software, das dependências das habilidades individuais e alteração dos requisitos pelo cliente. Os gestores dos projectos devem estar atentos e entender o impacto dos riscos nos projectos e antecipar os mesmos. Quadro passos são aplicadas para a gestão do risco:

Identificar o risco. Este é o primeiro estágio da gestão do risco. Neste estágio a maior preocupação do grupo SQA é de descobrir as possibilidades do risco no projecto. Esta identificação é feita usando braintorming approach.

Análise do Risco: Durante o processo de análise de risco, cada risco identificada é considerado e prioridade de acordo com a probabilidade e a grau de seriedade dos mesmo. O julgamento de cada risco é feito após a analisado e classificação dos mesmo e imediatamente uma medida deve ser tomado de forma a corrigir.

Planificação do risco: o plano considera cada risco identificado. É a função do grupo SQA planificar estratégias para cada risco.

Monitorização do risco: As actividades da monitorização do risco inicia junto com o projecto. Este envolve avaliações regulares de cada risco identificado para ajudar na decisão de qual dos riscos está se tornando mais ou menos amigável e quais são os efeitos dos riscos terão feito alguma alteração.

Tarefas da Garantia de Qualidade de Software

O objectivo principal dos engenheiros de software é ajudar ao cliente a obter um produto final de alta qualidade. Para a produção desse software de qualidade recomenda-se um conjunto de acções que tratam da planificação, da supervisão, da manutenção de registos, da análise e de relatórios relativos à garantia da qualidade.

Preparar plano de Garantia de Qualidade de Software para o projeto.

Participar no desenvolvimento do projeto de software da descrição do processo.

Revisão de atividades de engenharia Software para verificar a conformidade com o processo de software definida.

Auditoria de software designados produtos de trabalho a fim de verificar a sua conformidade com os definidos como parte do processo de software.

Assegurar que quaisquer desvios no software ou produtos de trabalho são documentados e manipulados de acordo com um procedimento documentado.

Gravar qualquer evidência de abandono e relatórios para gerenciamento.

Metas de SQA

As acções de SQA descritas na seção das tarefas do SQA são realizadas para atingir um conjunto de metas pragmáticas detalhamento:

Qualidade dos requisitos. A correção, a completude e a consistência do modelo de requisitos terão forte influência sobre a qualidade de todos os produtos seguintes.

A SQA deve assegurar-se de que a equipe de software tenha revisto apropriadamente o modelo de requisitos para a obtenção de um alto nível de qualidade.

Qualidade do projeto. Todo elemento do modelo de projeto deve ser avaliado pela equipe de software para garantir que apresente alta qualidade e que o próprio projeto esteja de acordo com os requisitos. A SQA busca atributos do projeto que sejam indicadores de qualidade.

Qualidade do código. O código-fonte e os produtos relacionados devem estar em conformidade com os padrões locais de codificação e apresentar características que irão facilitar a manutenção. A SQA deve isolar esses atributos que permitem uma análise razoável da qualidade do código.

Eficácia do controle de qualidade. A equipe de software deve aplicar os recursos limitados

de forma a obter a maior probabilidade possível de atingir um resultado de alta qualidade.

A SQA analisa a alocação de recursos para revisões e realiza testes para verificar se eles

estão ou não sendo alocados da maneira mais efetiva.

Quadro resumo das metas de SQA

Pressupõe

Meta	Atributo	Métrica
Qualidade das necessidades	Ambiguidade	Número de modificadores ambíguos (por exemplo, muitos, grande, amigável)
	Completude	Número de TBA, TBD
	Compreensibilidade	Número de seções/subseções
	Volatilidade	Número de mudanças por requisito
		Tempo (por atividade) quando é solicitada a mudança
	Facilidade de atribuição	Número de requisitos não atribuíveis ao projeto/código
	Clareza do modelo	Número de modelos UML
		Número de páginas descritivas por modelo
		Número de erros UML
Qualidade do projeto	Integridade da arquitetura	Existência do modelo da arquitetura
	Completude dos componentes	Número de componentes que se atribui ao modelo da arquitetura
		Complexidade do projeto procedural
	Complexidade da interface	Número médio de cliques para chegar a uma função ou conteúdo típico
		Apropriabilidade do layout
	Padrões	Número de padrões usados
Qualidade do código	Complexidade	Complexidade ciclométrica
	Facilidade de manutenção	Fatores de projeto (Capítulo 8)
	Compreensibilidade	Porcentagem de comentários internos
		Convenções de atribuição de variáveis
	Reusabilidade	Porcentagem de componentes reutilizados
	Documentação	Índice de legibilidade
Eficiência do controle de qualidade	Alocação de recursos	Porcentagem de horas de pessoal por atividade

Tarefas de Garantia de Qualidade de Software

Que uma rigorosa sintaxe e semântica podem ser definidas para cada linguagem de programação;

Permite o uso de uma abordagem rigorosa para a especificação dos requisitos de software;

Aplica técnicas matemáticas prova de exatidão para demonstrar que um programa está em conformidade com as suas especificações.

Conclusão

Nesta actividade visa discutiu-se os principais elementos de garantia da qualidade de software que é composta por uma série de tarefas associadas a dois elementos distintos — os engenheiros de software que realizam o trabalho técnico e um grupo de SQA que tem a responsabilidade pela planificação, supervisão, manutenção de registros, análise e relatórios referentes à garantia da qualidade.

Avaliação

Enumere o elemento de garantia de software.

O que e entende por grupo de SQA?

Quais são as Tarefas de Garantia de Qualidade de Software

Actividade 1.3 - Estatística de Garantia de Qualidade

Introdução

Depois de conhecer os elementos que garantem a qualidade de software, nesta actividade iremos aprender a estatística de garantia de qualidade como um instrumento quantitativo da analise da qualidade.

Detalhes da atividade

Estatística de Garantia de Qualidade

A estatística da garantia de qualidade reflete uma tendência crescente em toda a indústria de software para tornar mais quantitativa a análise da qualidade. Para software, a estatística da garantia da qualidade implica as seguintes etapas:

Informações sobre defeitos de software são coletados e categorizados

Cada defeito é rastreada até sua causa

Usar o Princípio de Pareto (80% dos defeitos pode ser rastreado até 20% das causas) isolar o "vital poucos" causas alternativo;

Mover para corrigir os problemas que causou a defeitos no "vital poucos"

Seis Sigma Engenharia de Software

é uma metodologia rigorosa e disciplinada que usa análise estatística e de dados para medir e melhorar o desempenho operacional de uma empresa através da identificação e da eliminação de defeitos em processos de fabricação e relacionados a serviços". O termo Seis Sigma é derivado de seis desvios-padrão — 3,4 ocorrências (defeitos) por milhão — implicando em um padrão de qualidade extremamente elevado.

A metodologia Seis Sigma define três etapas essenciais:

Definir os requisitos do cliente, resultados e objetivos do projeto através de métodos bem definidos de comunicação com o cliente.

Meça cada processo existente e sua saída para determinar o desempenho da qualidade atual (por exemplo calcular métricas de defeito)

Analisar métricas de defeito e determinar algumas causas virais.

Para um processo existente que precisa de melhoria

Melhore o processo, eliminando as causas de raiz para defeitos Controlar futuros trabalhos a fim de garantir que o trabalho futuro não reintroduzir as causas de defeitos

Se novos processos estão sendo desenvolvidos Projectar cada novo processo para evitar as causas raiz de defeitos e de ir ao encontro dos requisitos do cliente

Verifique se o modelo de processo irá evitar defeitos e atender aos requisitos do cliente

Confiabilidade do software

Definido como a probabilidade de falha operação livre de um programa de computador em um determinado ambiente para um período de tempo especificado

Pode ser medido diretamente e estimada usando dados históricos e de desenvolvimento (ao contrário de muitos outros factores de qualidade de software)

Problemas de confiabilidade do software geralmente pode ser rastreado de volta a erros no projeto ou implementação.

Medidas de Confiabilidade

Tempo médio entre falhas (MTBF) = MTTF + MTTR

MTTF = tempo médio até a falha

Tmdr = tempo médio de reparação

Disponibilidade = [MTTF / (MTTF + MTTR)] x 100%

Software de segurança

Definida como uma atividade de garantia de qualidade do software que se concentra na identificação de potenciais perigos que podem causar a falha de um sistema de software.

A identificação precoce de riscos permite que os desenvolvedores de software para especificar os recursos de design que pode eliminar ou pelo menos controlar o impacto de potenciais perigos.

Confiabilidade do software envolve determinar a probabilidade de que uma falha ocorrerá, enquanto a segurança de software analisa os modos em que falhas podem resultar em condições que podem levar a um incidente.

Normas de Qualidade ISO 9000

Os sistemas de garantia de qualidade são definidas como a estrutura organizacional, as responsabilidades, procedimentos, processos e recursos destinados à aplicação da gestão da qualidade.

ISO 9000 descreve os elementos de qualidade que devem estar presentes para que um sistema de garantia de qualidade para ser compatível com o padrão, mas não descrever como uma organização deve implementar estes elementos.

ISO 9001:2000 é o padrão de qualidade que contém 20 requisitos que devem estar presentes em um eficaz sistema de garantia de qualidade do software.

Plano de SQA

A secção de Gestão - descreve o lugar de SQA na estrutura da organização

Seção de documentação - descreve cada produto do trabalho produzida como parte do processo de software

As normas, práticas e convenções - seção lista todos os padrões aplicáveis/práticas aplicadas durante o processo de software e métricas de qualquer ser coletadas como parte do trabalho de engenharia de software

Revisões e auditorias - seção fornece uma visão geral da abordagem utilizada no revisões e auditorias para ser realizado durante o projeto

Seção de teste - referências o plano de teste e procedimento de documento e define os requisitos de manutenção de registro de teste

O relatório de problemas e acção corretiva secção - define procedimentos de reporte, rastreamento e resolver erros ou defeitos, identifica as responsabilidades organizacionais para estas actividades

Outras ferramentas, métodos de SQA, controle de alteração de registos, treinamento e gerenciamento de risco

Tmdr = tempo médio de reparação

Disponibilidade = [MTTF / (MTTF + MTTR)] x 100%

Software de segurança

Definida como uma atividade de garantia de qualidade do software que se concentra na identificação de potenciais perigos que podem causar a falha de um sistema de software.

A identificação precoce de riscos permite que os desenvolvedores de software para especificar os recursos de design que pode eliminar ou pelo menos controlar o impacto de potenciais perigos.

Confiabilidade do software envolve determinar a probabilidade de que uma falha ocorrerá, enquanto a segurança de software analisa os modos em que falhas podem resultar em condições que podem levar a um incidente.

Normas de Qualidade ISO 9000

Os sistemas de garantia de qualidade são definidas como a estrutura organizacional, as responsabilidades, procedimentos, processos e recursos destinados à aplicação da gestão da qualidade.

ISO 9000 descreve os elementos de qualidade que devem estar presentes para que um sistema de garantia de qualidade para ser compatível com o padrão, mas não descrever como uma organização deve implementar estes elementos.

ISO 9001:2000 é o padrão de qualidade que contém 20 requisitos que devem estar presentes em um eficaz sistema de garantia de qualidade do software.

Plano de SQA

A secção de Gestão - descreve o lugar de SQA na estrutura da organização

Secção de documentação - descreve cada produto do trabalho produzida como parte do processo de software

As normas, práticas e convenções - seção lista todos os padrões aplicáveis/práticas aplicadas durante o processo de software e métricas de qualquer ser coletadas como parte do trabalho de engenharia de software

Revisões e auditorias - seção fornece uma visão geral da abordagem utilizada no revisões e auditorias para ser realizado durante o projeto

Seção de teste - referências o plano de teste e procedimento de documento e define os requisitos de manutenção de registro de teste

O relatório de problemas e ação corretiva secção - define procedimentos de reporte, rastreamento e resolver erros ou defeitos, identifica as responsabilidades organizacionais para estas actividades

Outras - ferramentas, métodos de SQA, controle de alteração de registos, treinamento e gerenciamento de risco

Conclusão

A estatística da garantia de qualidade reflete uma tendência crescente em toda a indústria de software para tornar mais quantitativa a análise da qualidade. Para software, a estatística da garantia da qualidade implica as seguintes etapas:

Avaliação

Em que consiste a estatística de garantia de software?

Fale das seis sigma engenharia de software.

O que significa plano de qualidade.

Resumo da Unidade

Garantia da Qualidade é Conjunto de acções sistematizadas necessárias e suficientes para proveer confiança de que um produto ou serviço irá satisfazer os requisitos definidos da qualidade que, por sua vez, devem refletir as necessidades e as expectativas implícitas e explícitas do cliente. As actividades de garantia da qualidade de software são focadas na prevenção de defeitos e problemas, que podem surgir nos produtos de trabalho. Definição de padrões, metodologias, técnicas e ferramentas de apoio ao desenvolvimento fazem parte deste contexto. Duma forma generalizada, esta unidade discute apenas a questão relacionados a SQA.

Avaliação da unidade

Testa a sua compreensão!

instruções

Responda com clareza a questões a seguir:

Cada questão equivale a 5 pontos.

Avaliação

1. Fale detalhadamente sobre atributos de Qualidade de Software
2. Debruce sobre tarefas da Garantia de Qualidade de Software
3. O que percebe por normas de Qualidade ISO 9000
4. Fale sobre o plano SQA.

Resolução das Questões das unidades:

Actividade 1.1

1. A Garantia da Qualidade de Software (GQS) é a área-chave de processo do CMM cujo objetivo é fornecer aos vários níveis de gerência a adequada visibilidade dos projetos, dos processos de desenvolvimento e dos produtos gerados. A GQS atua como "guardiã", fornecendo um retrato do uso do Processo e não é responsável por executar testes de software ou inspeção em artefatos. Obtendo a visibilidade desejada, a gerência pode atuar de forma pontual no sentido de atingir os quatro grandes objetivos de um projeto de desenvolvimento de software, quais sejam, desenvolver software de alta qualidade, ter alta produtividade da equipe de desenvolvimento, cumprir o cronograma estabelecido junto ao cliente e não necessitar de recursos adicionais não previstos.Para conseguir esses objetivos a área-chave de processo GQS estimula a atuação das equipes responsáveis pelo desenvolvimento de software em diversas frentes objetivando internalizar comportamentos e ações, podendo-se destacar:

O planejamento do projeto e o acompanhamento de resultados;

O uso dos métodos e ferramentas padronizadas na organização;

A adoção de revisões técnicas formais;

O estabelecimento e a monitoração de estratégias de testes;

A revisão dos artefatos produzidos pelo processo de desenvolvimento;

A busca de conformidade com os padrões de desenvolvimento de software;

A implantação de medições associadas a projeto, processo e produto;

A utilização de mecanismos adequados de armazenamento e recuperação de dados relativos a projetos, processos e produtos;

A busca de uma melhoria contínua no processo de desenvolvimento de software.

2.

a) Qualidade é um conceito subjetivo, é o modo de ser, é a propriedade de qualificar os mais diversos serviços, objetos, indivíduos etc. Do latim qualitate. Qualidade está relacionado às percepções de cada indivíduo e diversos fatores como cultura, produto ou serviço prestado. Necessidades e expectativas influenciam diretamente nesta definição.

b) A qualidade de software é uma área de conhecimento da engenharia de software que objetiva garantir a qualidade do software através da definição e normatização de processos de desenvolvimento. Apesar dos modelos aplicados na garantia da qualidade de software atuarem principalmente no processo, o principal objetivo é garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado inicialmente.

3. O controle da qualidade é definido como um processo e métodos usados para monitorar o trabalho e observar se os requisitos estão sendo satisfeitos. O foco é justamente em revisões e remoção de defeitos antes mesmo do envio dos produtos. O controle da qualidade deve ser de responsabilidade da unidade organizacional que produz o produto. No entanto, é possível ter o mesmo grupo que constrói o produto, que realize também o controle da qualidade, ou estabelecer um grupo de controle da qualidade separado ou departamento dentro da mesma unidade organizacional que desenvolve o produto.

O controle da qualidade consistem de checklists bem definidos em um produto que é especificado no plano de garantia da qualidade. Um exemplos clássico de controle da qualidade são as inspeções de software. A inspeção é o grau mais maduro e formal dentro das revisões, sendo necessária uma preparação prévia, participantes definidos adequadamente e critérios de entrado e saída bem definidos.

O controle da qualidade é projetado para detectar defeitos e corrigir esses defeitos encontrados, enquanto que a garantida da qualidade é orientada através da prevenção de defeitos.

Actividade 1.2

1. Os elementos de garantia de qualidade de software são todas técnicas qualitativas usadas no processo no seu todo. Eles podem ser sistematizados de seguinte forma: Padrões estabelecidos pelas organizações tais como ISO, IEEE cujo a sua missão é de assegurar que as normas sejam adotadas e seguidas. A Revisões e auditorias estas são actividades de controle de qualidade realizadas por engenheiros de software com intuito de resolver erros e assegurar-se que as directrizes de qualidade são seguidos.

Testes – têm a função de controle de qualidade com o objectivo principal de detectar erros. Outro elemento fundamental é Coleta de análise de erro/defeito que tem a função de recolher e analisar dados erro e defeito para entender melhor como os erros são introduzidos e podem ser iliminados. Fora estes elementos existem como gerenciamento de mudanças, educação, gerencia dos fornecedores, administração e segurança, proteção e gestão de risco.

2. Grupo estabelecido e designado como responsável pela definição, manutenção e melhoria do processo de software da organização.

É o ponto focal de uma organização para software atividades da melhoria processo.
Os colaboradores pertencentes a esse grupo executam avaliações da potencialidade organizacional, desenvolvem planos para executar melhorias necessárias, coordenam a execução destes planos, e medem a eficácia destes esforços.

Para que um grupo de SQA seja bem sucedido requer habilidades e o conhecimento especializados da parte externa de muitas áreas fora da tecnologia de programação.

Principais Atividades:

Definir processo

Definir customizações do processo para os projetos

Manter o processo

Promover avaliações sobre o processo

Medir o processo

Inserir novas tecnologias

Prover treinamentos no processo

3. Que uma rigorosa sintaxe e semântica podem ser definidas para cada linguagem de programação, permite o uso de uma abordagem rigorosa para a especificação dos requisitos de software, aplica técnicas matemáticas prova de exatidão para demonstrar que um programa está em conformidade com as suas especificações.

Actividade 1.3

1. A estatística da garantia de qualidade reflete uma tendência crescente em toda a indústria de software para tornar mais quantitativa a análise da qualidade. Para software, a estatística da garantia da qualidade implica as seguintes etapas:

Informações sobre defeitos de software são coletados e categorizados

Cada defeito é rastreada até sua causa

Usar o Princípio de Pareto (80% dos defeitos pode ser rastreado até 20% das causas) isolar o vital poucos causas alternativo;

Mover para corrigir os problemas que causou a defeitos no vital poucos

2. É uma metodologia rigorosa e disciplinada que usa análise estatística e de dados para medir e melhorar o desempenho operacional de uma empresa através da identificação e da eliminação de defeitos em processos de fabricação e relacionados a serviços". O termo Seis Sigma é derivado de seis desvios - padrão — 3,4 ocorrências (defeitos) por milhão — implicando em um padrão de qualidade extremamente elevado.

A metodologia Seis Sigma define três etapas essenciais:

- Definir os requisitos do cliente, resultados e objetivos do projeto através de métodos bem definidos de comunicação com o cliente.
- Meça cada processo existente e sua saída para determinar o desempenho da qualidade atual (por exemplo calcular métricas de defeito)
- Analisar métricas de defeito e determinar algumas causas virais.
- Para um processo existente que precisa de melhoria
- Melhore o processo, eliminando as causas de raiz para defeitos Controlar futuros trabalhos a fim de garantir que o trabalho futuro não reintroduzir as causas de defeitos
- Se novos processos estão sendo desenvolvidos Projectar cada novo processo para evitar as causas raiz de defeitos e de ir ao encontro dos requisitos do cliente
- Verifique se o modelo de processo irá evitar defeitos e atender aos requisitos do cliente

Leituras e outros Recursos

Belgamo, A. & Martins, L.E.G., 2000. Estudo Comparativo sobre as técnicas de Elicitação de Requisitos do Software. In XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná.

Guerra, A.C. & Colombo, R.M.T., 2008. Qualidade de Produto de Software. PBQP/MCT. Dispon?vel em:< http://www. mct. gov. br/index. php/content/view/2867. html# lista>. Acesso em, 13(09), p.2009.

Jalote, P., 2008. A Concise Introduction to Software Engineering Sétima Edição. Springer, ed.,

Nazareth, D., 1999. Assembling a Metrics Suite for Rule-Based Systems Development. AMCIS 1999 Proceedings, p.23.

Pressman, R.S., 2011. Engenharia de Software - Uma Abordagem Profissional Sétima Edição. M. G. Hill, ed.,

Rocha Balthazar, G. da, 1981. Visão Geral da Qualidade de Software. Revista Eletrônica da Faculdade Metodista Granbery-http://re. granbery. edu. br-ISSN, p.0377.

Santos Bueno, C. de F. dos & Campelo, G.B., Qualidade de Software.

Gerard O'Regan, Introduction to Software Quality, Springer, 2014

Jeff Tian, Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement , IEEE Computer Society, 2005

Darrel Ince, An Introduction to Software Quality Assurance and Its Implementation, McGraw-Hill, 1994 https://en.wikipedia.org/wiki/Problem_frames_approach[Accessed 23/02/2016]

Alistair Sutcliffe, User-Centred Requirements Engineering, Springer-Verlag London Limited 2002 https://en.wikipedia.org/wiki/Requirements_analysis [Accessed 23/02/2016]

David J. Gilmore, Russel L. Winder and Francoise Detienne, User-Centred Requirements for Software Engineering Environments, Springer-Verlag Berlin Heidelberg GmbH, 1994

http://www.rbcs-us.com/documents/Defining-Testing-Article.pdf [accessed 24/02/2016]

http://www.sqa.org.uk/e-learning/SDPL03CD/page_16.htm[accessed 24/02/206]

http://searchsoftwarequality.techtarget.com/definition/conformance-testing[accessed [24/02/2016]

Paul Ammann and Jeff Offutt, INTRODUCTION TO SOFTWARE TESTING, CAMBRIDGE UNIVERSITY PRESS, 2008

William E. Perry, Effective Methods for Software Testing, Third Edition, Wiley Publishing, Inc., Indianapolis, Indiana, 2006 http://www.softwaretestinghelp.com/what-is-conformance-testing/ [accessed 24/02/2016] https://www.smashingmagazine.com/2010/06/design-better-faster-with-rapid-prototyping/[accessed 24/02/2016]

Unidade 2. - Análise de requisitos: Elicitação de requisitos e Desenvolvimento Centrado no Usuário

Introdução à Unidade

Esta unidade aborda principais elementos da engenharia de requisitos com enfoque na Elicitação de requisitos, Análise de requisitos e Desenvolvimento Centrado no usuário (DCU). Essa abordagem detalhista consiste em descrever cada elemento e apresentar as suas potencialidades com finalidade de desenvolver competências ao aluno no concernente ao desenvolvimento de sistema com qualidade.

Objetivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Dominar as técnicas de elicitação de requisitos;
- Destituir diferentes fases da engenharia de requisitos
- Aplicar na prática as técnicas de analise e elicitação;
- Dominar as diferentes metodologias de desenvolvimento centrados no cliente (DCU).

Termos-chave

Elicitação de requisitos:etapa que é responsável pela recolha de dados.

Análise de requisitos: Etapa que faz parte de todas as etapas do desenvolvimento.

Actividades de Aprendizagem

Actividade 1.1 - Elicitação de Requisitos

Introdução

A elicitação de Requisitos ou levantamento de requisitos é a base de todo o processo de engenharia de software. É nessa etapa que faz-se o levantamento de necessidades dos usuários e clientes, sistemas existentes na organização, regulamento, lei e outras informações relevantes do cliente. Trata-se de uma actividade complexa que que não se resume somente a inquire as pessoas o que elas desejam, mas sim analisar cuidadosamente a organização, o domínio da aplicação e os processos de negócio no qual o sistema será utilizado. Desta forma a elicitação combina elementos de resolução de problemas, elaboração, negociação e especificação.

Detalhes da actividade

Elicitação de Requisitos

Elicitar - significa descobrir, tornar explícito, obter o máximo de informações para o conhecimento do objeto em questão.

Actividades da elicitação

A figura abaixo apresenta as actividades do levantamento do requisitos de uma forma sequencial.

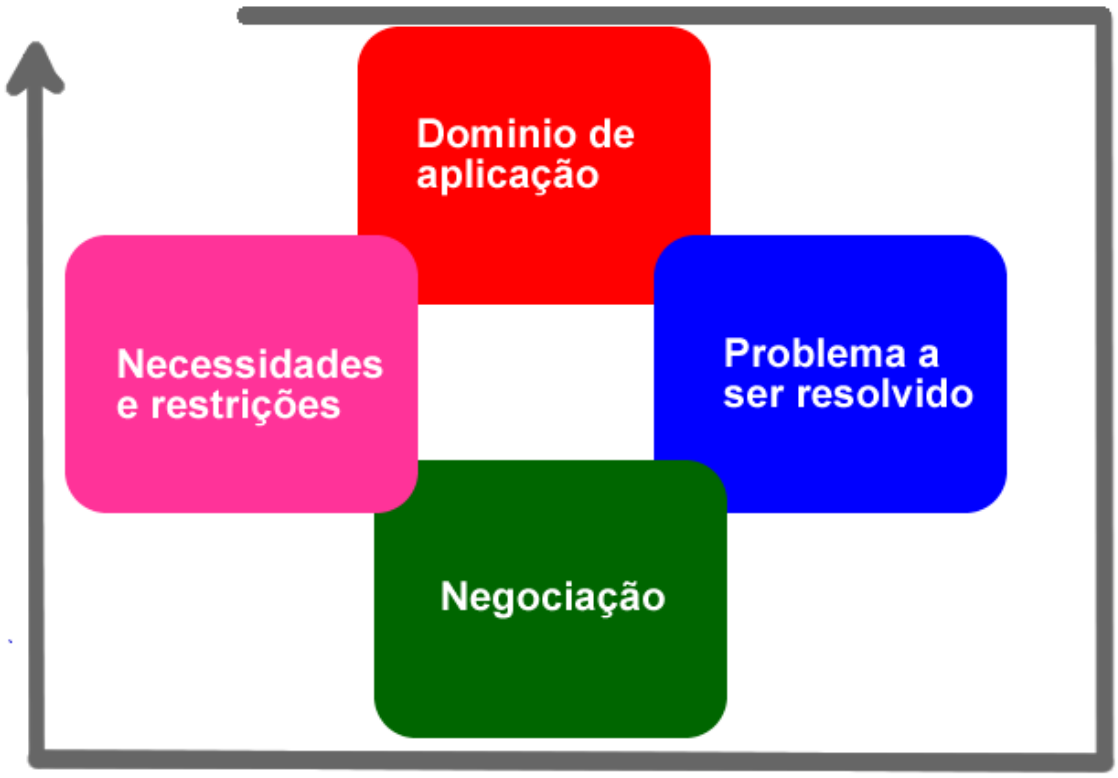


Fig.1. Levantamento de requisitos, Fonte:(Autor)

- Entendimento do domínio da aplicação: Conhecimento geral da local na qual o sistema será aplicado;
- Entendimento do problema: Entendimento dos detalhes do problema específico a ser resolvido com o auxílio do sistema a ser desenvolvido;
- Entendimento do negócio: entendimento de como o sistemas interagem e contribuem de forma geral com o objectivo do negocio do cliente;
- Entendimento das necessidades e das restrições dos interessados: Entender a demanda de apoio para a realização do trabalho das partes no sistema, entender os processos de trabalho a serem apoiados pelo sistema.

Problemas de levantamento de requisitos

- Usuários podem não ter uma idéia precisa do sistema por eles requerido;
- Usuários têm dificuldades para descrever seu conhecimento sobre o domínio do problema;
- Usuários e Analistas têm diferentes pontos de vista do problema ;
- Usuários podem não familiarizar-se com o novo sistema e escusa-se a participar da elicitação ou mesmo fornecer informação não fiável.
- O processo da elicitação de requisitos

É um processo iterativo, com uma contínua validação de uma atividade para outra, a figura abaixo apresenta o ciclo completo desse processo:

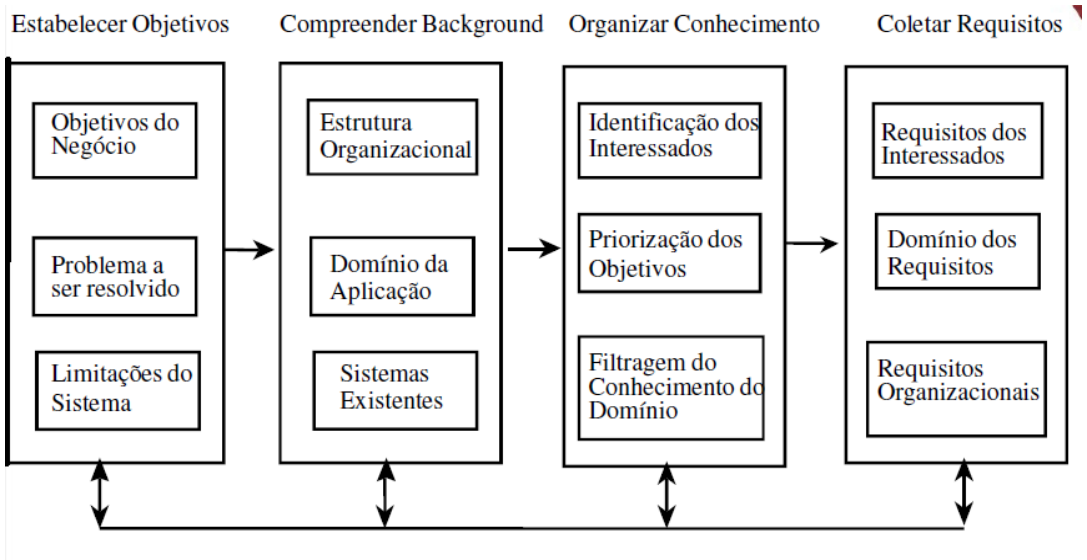


Figura: Ciclo completo do processo

Fonte: (Jaelson Castro, 2013)

Estágios da Elicitação

Definição dos objectivos

Os objectivos organizacionais devem ser estabelecidos incluindo objetivos gerais do negócio, uma descrição geral do problema a ser resolvidos porque o sistema é necessário e as limitações do sistema.

Aquisição de conhecimento do background

Informação de background do sistema inclui informação acerca da organização onde o sistema será instalado, o domínio de aplicação do sistema e informação acerca de outros sistemas existente

Organização do conhecimento

A grande quantidade de dados recolhido nos estágios anteriores devem ser organizadas e colocadas em ordem.

Recolha de requisitos dos stakeholders

Os stakeholders do sistema são consultados para descoberta de seus requisitos.

Técnicas de Elicitação

Segundo Belgamo & Martins, 2000, Para uma boa elicitação de requisitos, existem técnicas para o melhor entendimento e comunicação entre clientes e analistas, para que problemas, não ocorrem, ou se ocorrem, que sejam mais facilmente resolvidos. A tabela abaixo apresenta o resumo das principais técnicas de elicitação:

Categorias	Técnica
Conversação	• Entrevistas; • Workshops; • Brainstorming(debates); • Questionários e; • Grupo focal.
Observação	• Etnografia; • Observação e; • Protocolo de Análise.
Análitico	• Reuso de Requisitos; • Estudo de Documentação e; • Laddering.
Sintético	• Sessões de JAD/RAD e; • Prototipação.

Tabela 1. Técnicas de elicitação

Na conversação é muito importante estar atento, porque uma vez não percebida a informação que o cliente pretende dar, isso pode criar problemas em todo o projecto, como por exemplo, normalmente a entrevista é a primeira técnica utilizada para descobrir as necessidades dos usuários. Agora é o momento de escutar mais do que falar. Para os analistas iniciantes, o início das entrevistas ode ser um pouco confuso, pois muitas informações são despejadas de uma só vez, sem organização. O cliente está falando de determinado assunto, de repente lembra-se de outra funcionalidade, para então retornar ao assunto inicial. Desta forma é preciso usar diferentes ferramentas como gravadores para reter a informação e escutar mais vezes.

No analítico por no exemplo de documentação, a documentação: descreva o problema de forma clara e objetiva. Em caso de dúvidas, consulte o cliente e evite inferências. Procure usar exemplos citados pelas partes interessadas (stakeholders). A adoção de diagramas e figuras sempre ajuda na documentação e entendimento dos requisitos. A criação de protótipos também contribui para o entendimento comum da solução proposta.

Conclusão

A elicitação de Requisitos é a base de todo o processo de engenharia de software. É nessa etapa que faz-se o levantamento de necessidades dos usuários e clientes, sistemas existentes na organização, regulamento, lei e outras informações relevantes do cliente. Trata-se de uma actividade complexa que que não se resume somente a inquirir as pessoas o que elas desejam, mas sim analisar cuidadosamente a organização, o domínio da aplicação e os processos de negócio no qual o sistema será utilizado. Desta forma a elicitação combina elementos de resolução de problemas, elaboração, negociação e especificação.

Avaliação

Com base nos conhecimentos dados, explica na totalidade o processo de levantamento de requisitos.

Fale das técnicas de elicitação estudadas.

Actividade 1.2 - Análise de Requisitos

Introdução

A análise de requisitos é uma etapa crucial no desenvolvimento de software, ela aparece logo após a elicitação de requisitos que ditam as especificação de características operacionais do software. Tem como objectivo principal indicar a interface do software com outros elementos do sistema e estabelecer restrições que software deve fazer. Permite elaboração de mais necessidades básicas estabelecidas durante as etapas anteriores da engenharia de requisitos

Esta, estabelece também um conjunto acordado de requisitos consistentes e sem ambiguidades, que possa ser usado como base para o desenvolvimento do software.

Para tal, diversos tipos de modelos são construídos, e para garantir o DCU - Desenvolvimento Centrado no Usuário, a análise de requisitos inclui também a negociação para resolver conflitos detectados.

Detalhes da atividade

A análise de sistemas é uma actividade de construção de modelos e por sua vez um modelo é uma representação de alguma coisa do mundo real, uma abstração da realidade, e, portanto, representa uma seleção de características do mundo real relevantes para o propósito do sistema em questão.

Os modelos são construídos para possibilitar o estudo do comportamento do sistema, facilitar a comunicação entre os componentes da equipe de desenvolvimento e clientes e usuários, garantir a discussão de correções e modificações com o usuário e formar a documentação do sistema.

Nível de abstração

Existem três níveis de modelos:

Conceitual: considera características do sistema independentes do ambiente computacional no qual o sistema será implementado. Os Modelos conceituais são construídos na actividade de análise de requisitos.

Lógico: características dependentes de um determinado tipo de sistema computacional. Essas características são, contudo, independentes de produtos específicos. Estes, modelos são típicos da fase de projeto.

Físico: características dependentes de um sistema computacional específico, isto é, uma linguagem e um compilador específico. Esses modelos podem ser desenhados tanto na fase de projeto quanto na fase de implementação.

Nas primeiras etapas do desenvolvimento do software, o engenheiro de software representa o sistema através de modelos conceituais e nas etapas subsequentes (a lógica e física) são representados novos modelos que por sua vez dão a um projetista de software informações que podem ser traduzidas em projectos de arquitetura, de interfaces e de componentes.

No entanto segundo o Pressman (2011), a acção de modelagem resulta em um ou mais modelos a contar:

" Modelos baseados em cenários de requisitos do ponto de vista de vários "atores" do sistema;

Modelos de dados que representam o domínio de informações para o problema;

Modelos orientados a classes que representam classes orientadas a objetos e a maneira por meio da qual as classes colaboram para atender aos requisitos do sistema;

Modelos orientados a fluxos que representam os elementos funcionais do sistema e como eles transformam os dados à medida que percorrem o sistema;

Modelos comportamentais que representam como o software se comporta em consequência de "eventos" externos." (Preman, 2011)

Com o avanço das técnicas de modelagem, atualmente o modelo mais usado é a conceitual devido as qualidade que ela apresenta como a modelagem de classes que é uma representação de classes orientadas a objectos e as colaborações resultantes que permitem que um sistema funcione.

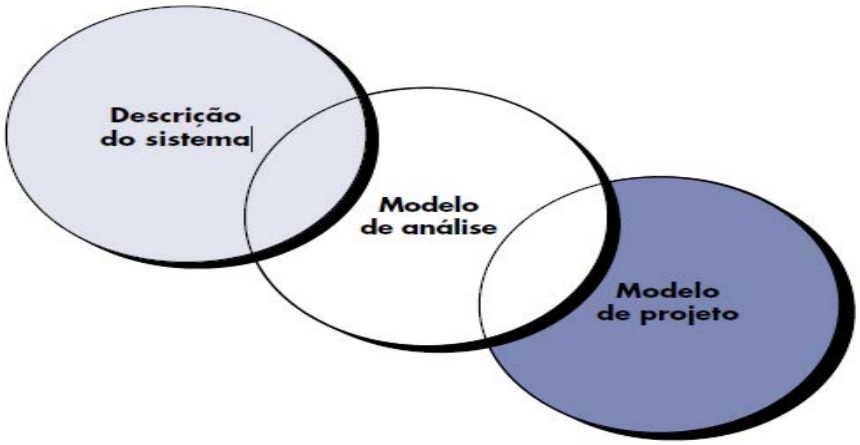


Figura 2: Processo de Modelagem

Fonte: (Pressman, 2006)

O modelo de análise o analista de sistema concentra-se no usuário tentando responder diversas respostas tais como, que interação com usuário ocorre em dadas circunstância, os objectos o sistema deve manipular, as funções que o sistema deve executar, que interfaces são definidas e as respectivas restrições.

Segundo o Pressman, 2006 este modelo deve alcançar três objetivos primários:

- Descrever o que o cliente solicita,
- Estabelecer uma base para a criação de um projeto de software e;
- Definir um conjunto de requisitos que possa ser validado assim que o software for construído.

"O modelo de análise preenche a lacuna entre uma descrição sistêmica que descreve o sistema como um todo ou a funcionalidade de negócio que é atingida aplicando-se software, hardware, dados, pessoal e outros elementos de sistema e um projeto de software que descreve a arquitetura, a interface do usuário e a estrutura em termos de componentes do software:

Analise de domínio

A Análise de Domínio é a atividade diretamente ligada à reutilização na Engenharia de

Requisitos. A Análise de Domínio visa capturar os elementos relevantes de um domínio de aplicações e disponibilizá-los para serem utilizados no desenvolvimento de diferentes sistemas de apoio a negócios neste domínio. Assim, a Análise de Domínio busca explicar e modelar aspectos de domínio, produzindo artefatos que contêm informações sobre o domínio e que podem ser reutilizados no desenvolvimento de sistemas.

A análise do domínio é uma actividade contínua da engenharia de software que não está ligada a nenhum projecto de software específico.

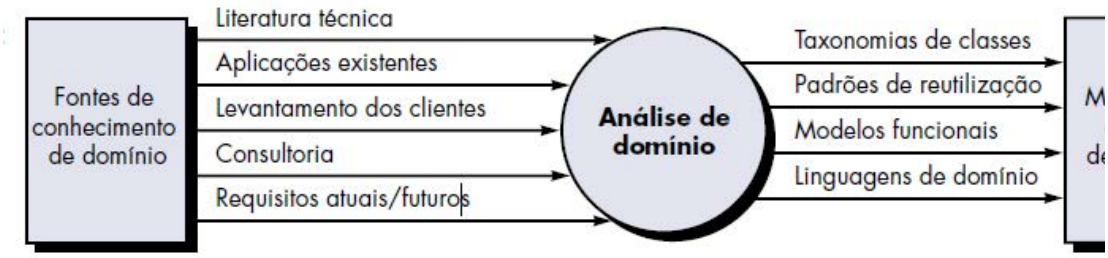


Figura 3: Analise de dominio

Abordagens à modelagem de requisitos

O modelo de requisitos apresenta duas abordagens: A análise estruturada e análise orientada a objetos. A estruturada considera os dados e os processos que transformam os dados em entidades separadas enquanto que a abordagem análise orientado a objectos concentra se na definição de classes e na maneira pela qual colaboram entre si para atender às necessidades dos clientes. A UML e o Processo Unificado são predominantemente orientados a objectos.

Diferença entre abordagens

Factor	Análise estruturada	Análise orientada a objecto
Ferramentas	Diagramas de Fluxo de dados, Diagramas de contexto, Modelo Entidade Relacionamento	Diagramas de Casos de Uso, Diagrama de Classes, E outros diagramas (Pacotes,actividades, etc) - Ferramentas UML
Foco	Processos	Objecto
Riscos	Elevado	Reduzido
Tipo de aplicações	Sistemas com requisitos claramente definidos; sistemas de pequeno e médio porte	Projectos de larga escala; sistemas complexos
Metodologias	SSADM, JAD, RAD	FDD, SCRUM, OOAD

Tabela 1: Abordagens de requisitos

Conclusão

O entendimento completo do sistema é obrigatório para o sucesso do processo do desenvolvimento do software, a fraca análise e a especificação do sistema pode incomodar os utilizadores proporcionar uma aflição aos desenvolvedores. A análise de requisitos é um processo que apura, analisa, documenta e verifica os requisitos. Este, inclui as especificações das características operacionais do sistema, a interface do sistema com outros elementos e estabelece restrições que o software deve adequar. A analise de requisitos e especificações proporcionam aos desenvolvedores e clientes um meio para avaliar a qualidade uma vez que o software é construído.

Avaliação

1. Os modelos são construídos para possibilitar o estudo do comportamento do sistema, facilitar a comunicação entre os componentes da equipe de desenvolvimento e clientes e usuários, garantir a discussão de correções e modificações com o usuário e formar a documentação do sistema. Estes modelos apresentam três níveis de abstração.

   a. Quais são? Fale detalhadamente de cada um deles.

2. O modelo de requisitos apresenta duas abordagens: A análise estruturada e análise orientada a objetos:

   a. Diferencie-os;

   b. Diferencie-os quanto as ferramentas, Foco, metodologias, risco e aplicação com detalhadamente.

Actividade 1.3 - Análise e Desenvolvimento Centrado no Usuário

Introdução

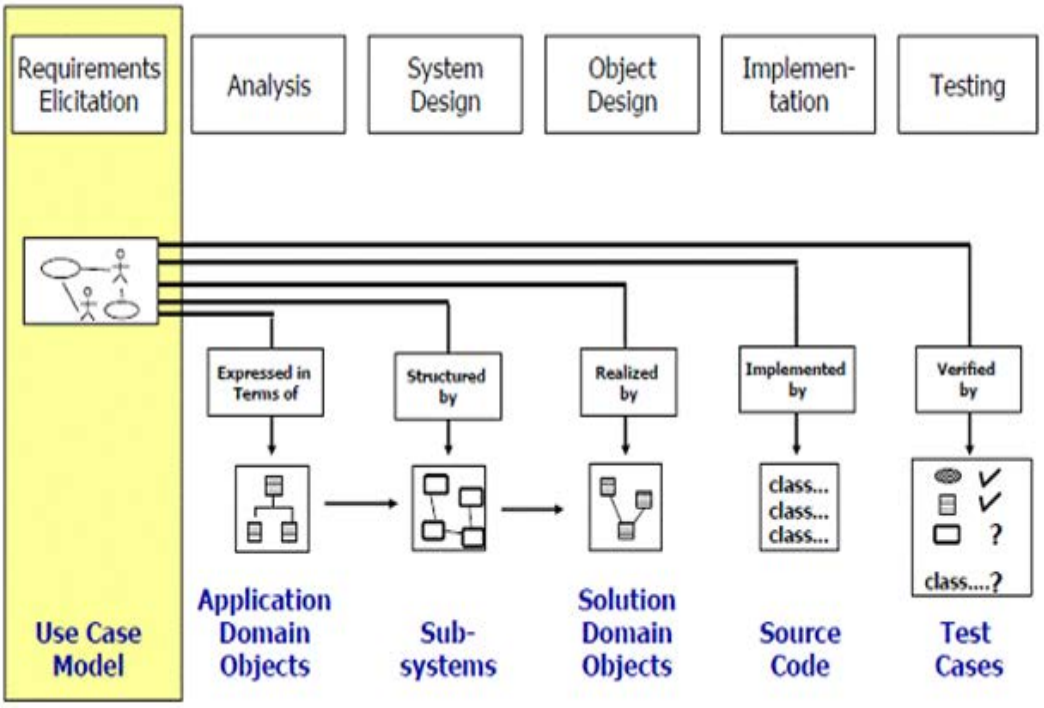
Requisito de indução engloba todas as actividades envolvidas em descobrir os requisitos de um sistema. Desenvolvedores de sistemas e engenheiros trabalham em estreita relação com o cliente e usuários finais (partes interessadas) para saber mais sobre o problema a ser resolvido, para descrever a funcionalidade do sistema, o desempenho do sistema, restrições de hardware e assim por diante

Detalhes da atividade

Requisito de indução não é apenas um simples processo de coleta de requisitos, mas um processo altamente complexo devido ao seguinte campo inevitável ambientes:

- O cliente raramente têm uma imagem clara das suas necessidades
- Pessoas diferentes têm requisitos conflitantes
- O ambiente de negócios na qual o processo de indução ocorre está mudando constantemente, daí podem desencadear alguns requisitos para alterar ou novos requisitos resultantes de novas partes interessadas.
- Muitos diferentes processos de indução não existem requisitos que um eliciador tem de escolher.
- Requisitos de indução é a primeira atividade do processo de desenvolvimento de software e é um processo de dentro como integro representado na figura 1.
- Observe também que cada atividade da fase de desenvolvimento do sucessor sistematicamente transforma a exigência de indução resultado do sucessor fase de desenvolvimento.

Figura 1: obrigação de Indução em um processo de desenvolvimento de software



Fonte: (internet)

A exigência de processo de indução gira em torno de quatro actividades básicas perspectivas;

O domínio de aplicativo, problema, os negócios e a perspectiva das partes interessadas. Aplicação perspectiva de domínio implica uma compreensão do conhecimento da área geral onde o sistema é aplicado. O problema perspectiva implica uma compreensão de detalhes do problema específico do cliente onde o sistema será aplicado. A perspectiva de negócios implica compreender a sistemas como interagir e contribuir para objectivos de negócio globais. As partes interessadas perspectivas financeiras implica compreender as necessidades e limitações do sistema de partes interessadas que é objectivo subir com as necessidades específicas das pessoas que necessitam de suporte do sistema no seu trabalho.

Visto a partir da perspectiva do processo, requisitos elicitação engloba quatro fases de indução de base que inclui: definição de objetivos, fundo de aquisição de conhecimentos e a organização do conhecimento e a exigência de partes interessadas coleção. Na definição de objectivos a preocupação é a de estabelecer objetivos organizacionais, global e resposta por que razão o sistema é necessário. O fundo de aquisição de conhecimentos a preocupação com a recolha e compreensão mais informações gerais sobre o sistema. A organização do conhecimento fase envolve a organizar, priorizando e estruturação da grande quantidade de dados coletados em fases anteriores. A fase de coleta de requisitos das partes interessadas visa envolver as partes interessadas do sistema para descobrir seus requisitos.

DCU - Design Centrado no utilizador

Até a década de 80, quando começaram os esforços para o estudo em Interação Homem-Computador (IHC), os profissionais das ciências da computação eram os únicos que se

envolviam nos processos de desenvolvimento de software, mas com a complexidade e o tamanho dos sistemas computadorizados atuais torna praticamente impossível de construí-lo sem organização. É necessária a utilização de técnicas de engenharia de software. Entre os diversos métodos utilizados, está um grupo que esta em evidência, as metodologias ágeis.

Design Centrado no Usuário implica em uma abordagem multidisciplinar ao design, baseada no envolvimento ativo dos usuários para um entendimento claro do papel do usuário, dos requisitos das tarefas e das iterações de design (como projeto e processo) e avaliações. É considerada como elemento-chave para utilidade e usabilidade. (MAO et al, 2001)

Uma abordagem de se trabalhar o desenvolvimento de qualquer produto a partir das necessidades dos usuários, Envolve a aplicação de metodologias de usabilidade essenciais ao desenvolvimento de sistemas interativos e produtos. Esta nova abordagem aplica as metodologias interativas como a prototipagem.

DCU apresenta quatro fases:

1. Análise
2. Design
3. Implementação
4. Deployment

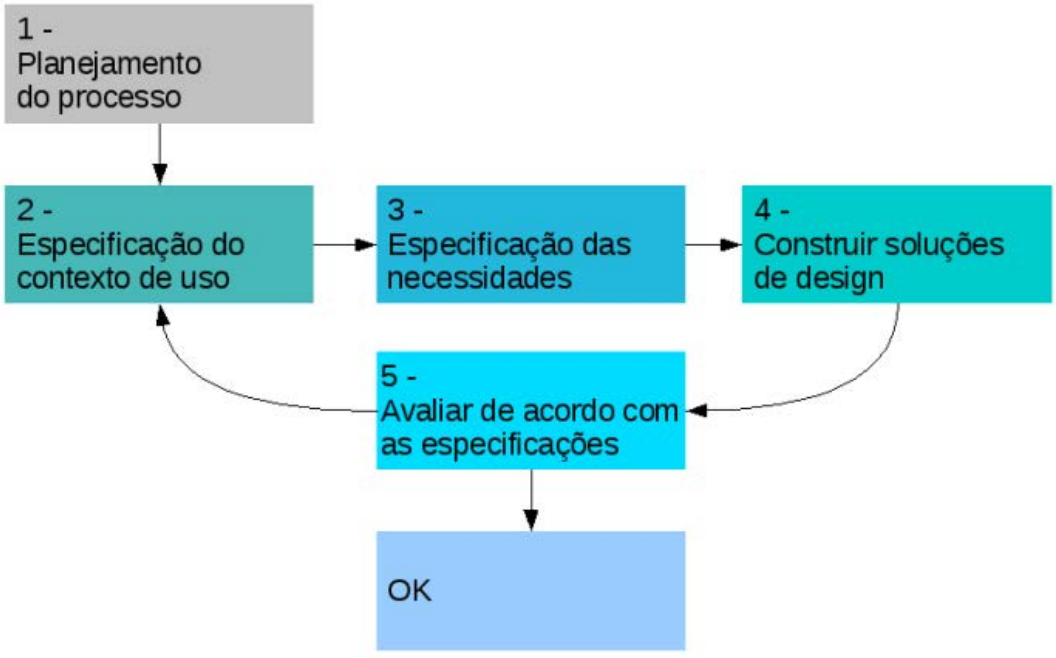


Figura: Fases de DCU

Fonte: (internet)

Fase de Análise:

- Como em outros modelos, a definição de abordagem é feita a partir do encontro com os envolvidos.
- Previsão de aplicação de métodos de usabilidade no projeto
- Montagem de equipe multidisciplinar
- Definição dos objetivos de usabilidade
- Conduzir estudos de campo
- Observar produtos dos competidores
- Criação de perfis de uso e desenvolver análises de tarefas
- Documentar cenários de uso
- Documentar requisitos de performance

Design

Brainstorming, Desenvolvimento de fluxos de navegação e organização de estrutura, Simulação de aplicação dos conceitos desenvolvidos, Prototipação de papel, Condução de testes em protótipos, Prototipação avançada, Condução de testes em protótipos, Definição de padrões e guias de orientação e Especificações de design.

Implementação

Realização de avaliações heurísticas, Trabalho conjunto com a equipe de execução, Realização de testes de usabilidade

Deployment

Buscar feedback dos usuários, Estudos de campo para obter informações de uso, Verificar objetivos e metas de usabilidade

Ler mais:

ttp://www.slideshare.net/karinedrumond/o-que-design-centrado-no-usurio

http://www.caiocesar.cc/wp/wp-content/uploads/2014/05/20080517_Interminas_CaioCesar_DCU.pdf

Principais técnicas

Entrevista com usuários e stakeholders

Observação em campo

Questionários

Card Sorting

Personas

Prototipação

Testes com usuário

Conclusão

Nesta actividade que estamos a descrever o requisito de indução como parte integrante do processo de desenvolvimento de software. Destacamos a sua complexidade devido à inevitável ocorrendo incidências no campo, actividades subjacentes e a lógica de etapas envolvidas ao requisito de indução é visto como um processo.

Avaliação

1. Utilizando os conhecimentos adquiridos a partir do curso de engenharia de software escolha um exemplo simples da aplicação de software do modelo de caso de uso e desenvolver sistematicamente a transformações associadas em:

a. Objetos de domínio de aplicativo
b. Subsistemas
c. Objetos de domínio de solução
d. Código fonte esqueleto
e. Casos de teste

Resumo da Unidade

Esta unidade apresentou o processo de análise de requisito de olhar sobre aspectos de elicitação de requisitos. Debruça também a cerca do DCU - Desenvolvimento Centrado no Utilizador através da abordagem de negociação presente em cada modelo do processo de desenvolvimento. A Elicitação de requisitos, análise e actividades de negociação foram descritos.

Avaliação da Unidade

Testa a sua compreensão.

Instruções:

Responda com clareza as questões.

O total de questões equivalem a 20 pontos. As questões 1,2 e 3 equivalem a 2 pontos cada. As restantes 3.5 pontos cada

Avaliação

1. O que entendes de elicitação de requisitos?
2. Estabeleça uma relação das etapas do ciclo do processo de elicitação de requisitos
3. O que são técnicas de elicitação de exemplo de duas em cada categoria descreva-as.
4. O que entende de analise de requisitos?
5. Qual é a primeira etapa do processo de engenharia de requisitos?
6. Avalie a análise do domínio.
7. Fale com profundidade sobre o design centrado no usuário, as suas etapas e técnicas.

Respostas da Unidade

Actividade 2.1

1. É um processo iterativo, com uma contínua validação de uma atividade para outra, a figura abaixo apresenta o ciclo completo desse processo:

Definição dos objectivos Os objectivos organizacionais devem ser estabelecidos incluindo objetivos gerais do negócio, uma descrição geral do problema a ser resolvidos porque o sistema é necessário e as limitações do sistema. Na Aquisição de conhecimento do background a Informação de background do sistema inclui informação acerca da organização onde o sistema será instalado, o domínio de aplicação do sistema e informação acerca de outros sistemas existente. A Organização do conhecimento a grande quantidade de dados recolhido nos estágios anteriores devem ser organizadas e colocadas em ordem. Recolha de requisitos dos stakeholders, as partes intervenientes do sistema são consultados para descoberta de seus requisitos.

2. As técnicas de levantamento de requisitos têm por objetivo superar as dificuldades relativas a esta fase. Todas as técnicas possuem um conceito próprio e suas respectivas vantagens e desvantagens, que podem ser utilizadas em conjunto pelo analista.

Para uma boa elicitação de requisitos, existem técnicas para o melhor entendimento e comunicação entre clientes e analistas, para que problemas, não ocorrem, ou se ocorrem, que sejam mais facilmente resolvidos. A tabela abaixo apresenta o resumo das principais técnicas de elicitação:

Categorias	Técnica
Conversação	Entrevistas; Workshops; Brainstorming(debates); Questionários e; Grupo focal.
Observação	Etnografia; Observação e; Protocolo de Análise.
Análitico	Reuso de Requisitos; Estudo de Documentação e; Laddering.
Sintético	Sessões de JAD/RAD e; Prototipação.

Actividade 2.2

1. a). Existem três modelos de abtracçaõ: A Conceitual, Logico e Físico.

Conceitual: considera características do sistema independentes do ambiente computacional no qual o sistema será implementado. Os Modelos conceituais são construídos na

actividade de análise de requisitos e o Lógico considera as características dependentes de um determinado tipo de sistema computacional. Essas características são, contudo, independentes de produtos específicos. Estes, modelos são típicos da fase de projeto enquanto que o Físico características dependentes de um sistema computacional específico, isto é, uma linguagem e um compilador específico. Esses modelos podem ser desenhados tanto na fase de projeto quanto na fase de implementação.

2. Resposta(a &b):

Análise Estruturada

A análise estruturada é uma atividade de construção de modelos. Utiliza uma notação que é particular ao método de análise estruturada para com a finalidade de retratar o fluxo e o conteúdo das informações utilizadas pelo sistema, dividir o sistema em partições funcionais e comportamentais e descrever a essência daquilo que será construído.

A análise estruturada é muito difícil de ser modelada e, rastrear e gerenciar mudanças manualmente. Por essas e outras razões, as ferramentas DFD tornaram-se uma abordagem preferida. Para uma pré-elaboração de um projeto de desenvolvimento de software.A análise estruturada também contém gráficos que possibilita o analista criar modelos defluxo de informação, com uma heurística para o uso dos símbolos, juntamente com um dicionário de dados, e narrativas de processamentos como o complemento aos modelos defluxo de informação. Um modelo de fluxo pode ser criado para qualquer sistema baseado em computador, independentemente do tamanho e complexidade.

O dicionário de dados é uma listagem organizada de todos os elementos de dados que são pertinentes ao sistema, com definições precisas e rigorosas, de forma que tanto o usuário como os analistas de sistemas tenham uma compreensão comum das tarefas, das saídas, dos componentes dos depósitos e até mesmo dos cálculos intermediários. Atualmente, isto é inserido quase sempre como parte de uma ferramenta de projeto e análise estruturada

Orientada a Objetos

A programação orientada a objeto é diferente da programação estruturada. Na programação orientada a objeto, funções e dados estão juntos, formando o objeto. Essa abordagem cria uma nova forma de analisar, projetar e desenvolver programas, é uma forma mais abstrata, e genérica, que permite um maior reaproveitamento dos códigos, e facilita a sua manutenção. Observe que a modelagem orientada a objeto, não é somente uma nova forma de programar, mas uma nova forma de pensar um problema, de forma abstrata, utilizando conceitos do mundo real e não conceitos computacionais. Na programação orientada a objeto o conceito de objeto deve acompanhar todo o ciclo de desenvolvimento do software. A POO também inclui uma nova notação e exige do analista/programador o conhecimento dessa notação (diagramas de classe, diagramas de interação, diagramas de sequência, etc.). Atualmente existem centenas de bibliotecas, cuidadosamente desenhadas, para dar suporte aos programadores menos sofisticados. Desta forma os programadores podem montar seus programas unindo as bibliotecas externas com alguns objetos que criaram, ou seja, poderão montar suas aplicações rapidamente, contando com módulos pré-fabricados.

O usuário final verá todos os ícones e janelas da tela como objetos e associará a manipulação desses objetos visuais à manipulação dos objetos reais que eles representam. Enxerga o mundo como objetos com estrutura de dados e comportamentos O objetivo é desenvolver uma série de modelos de análise, satisfazendo um conjunto de requisitos definidos pelo cliente O problema não está em aprender como programar em uma linguagem OO, mas sim em aprender a explorar as vantagens que as linguagens OO oferecem. Portanto, para o sucesso de um projeto OO é necessário seguir boas práticas de engenharia discutidas na literatura e pesquisando padrões já consolidados e aprovados.

Leituras e outros Recursos

- Belgamo, A. & Martins, L.E.G., 2000. Estudo Comparativo sobre as técnicas de Elicitação de Requisitos do Software. In XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná.
- Guerra, A.C. & Colombo, R.M.T., 2008. Qualidade de Produto de Software. PBQP/MCT. Disponıvel em:< http://www. mct. gov. br/index. php/content/view/2867. html# lista>. Acesso em, 13(09), p.2009.
- Jalote, P., 2008. A Concise Introduction to Software Engineering Sétima Edição. Springer, ed.,
- Nazareth, D., 1999. Assembling a Metrics Suite for Rule-Based Systems Development. AMCIS 1999 Proceedings, p.23.
- Pressman, R.S., 2011. Engenharia de Software - Uma Abordagem Profissional Sétima Edição. M. G. Hill, ed.,
- Rocha Balthazar, G. da, 1981. Visão Geral da Qualidade de Software. Revista Eletrônica da Faculdade Metodista Granbery-http://re. granbery. edu. br-ISSN, p.0377.
- Santos Bueno, C. de F. dos & Campelo, G.B., Qualidade de Software.
- Gerard O'Regan, Introduction to Software Quality, Springer, 2014
- Jeff Tian, Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement , IEEE Computer Society, 2005
- Darrel Ince, An Introduction to Software Quality Assurance and Its Implementation, McGraw-Hill, 1994
- https://en.wikipedia.org/wiki/Problem_frames_approach[Accessed 23/02/2016]
- Alistair Sutcliffe, User-Centred Requirements Engineering, Springer-Verlag London Limited 2002
- https://en.wikipedia.org/wiki/Requirements_analysis [Accessed 23/02/2016]
- David J. Gilmore, Russel L. Winder and Francoise Detienne, User-Centred Requirements for Software Engineering Environments, Springer-Verlag Berlin Heidelberg GmbH, 1994
- http://www.rbcs-us.com/documents/Defining-Testing-Article.pdf [accessed 24/02/2016]
- http://www.sqa.org.uk/e-learning/SDPL03CD/page_16.htm[accessed 24/02/206]
- http://searchsoftwarequality.techtarget.com/definition/

conformance-testing[accessed [24/02/2016]

- Paul Ammann and Jeff Offutt, INTRODUCTION TO SOFTWARE TESTING, CAMBRIDGE UNIVERSITY PRESS, 2008
- William E. Perry, Effective Methods for Software Testing, Third Edition, Wiley Publishing, Inc., Indianapolis, Indiana, 2006
- http://www.softwaretestinghelp.com/what-is-conformance-testing/[accessed 24/02/2016]
- https://www.smashingmagazine.com/2010/06/design-better-faster-with-rapid-prototyping/[accessed 24/02/2016]

Unidade 3. Verificação e validação.

Introdução à Unidade

Verificação e Validação (V&V) é um processo de verificação e análise contínua, onde as actividades decorrem em cada fase do processo de software, ele abrange toda as etapas de produção iniciando com as análise de requisitos até os testes de produtos. Estes processos não são equivalentes. A Validação preocupa-se com o desenvolvimento correto do produto final a verificação preocupa-se com as etapas do desenvolvimento do mesmo de uma forma correta.

Desta forma, o papel da verificação nos remete-nos a faz nos chegar a conclusão de que estamos no caminho certo no concernente a produção de acordo com as especificações e se entende ou não os requisitos funcionais e não funcionais. Segundo o Pressman (2006), ele define a verificação e validação da seguinte forma:

" Verificação refere-se ao conjunto de tarefas que garantem que o software implementa corretamente uma função específica. Validação refere-se a um conjunto de tarefas que asseguram que o software foi criado e pode ser rastreado segundo os requisitos do cliente." o objectivo principal do V&V é de: Introduzir verificação e validação de software. Descrever o processo de inspeção de programa Explicar análise estática. Introduzir o processo de desenvolvimento de software através de teste.

Este processo de V&V contém duas abordagens para a verificação e analise de sistema, onde a primeira seria a inspeção de software ou revisões por pares também chamado de técnica de V&V estática, onde analisam e verificam as representações do sistema como um documento de requisitos, diagramas de projeto, código-fonte do programas, etc. e a segunda abordagem seria a de testes de software também conhecida com técnica de V&V dinâmica, que envolvem a execução do software com dados de testes a fim de checar as saídas do software, o comportamento operacional no intuito de verificar se o seu desempenho está conforme o desejado.

Objetivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Compreender o conceito validação e suas técnicas;
- Saber aplicar as estratégias dos testes na prática;
- Demostrar o funcionamento da prototipagem na validação de projectos.

Termos-chave

Validação: Verificar que o programa implementado satisfaz as expectativas do cliente

Verificação: Verificar que o programa está em conformidade com as suas especificações

Actividades de Aprendizagem

Actividade 1.1 - Definição da Missão de teste

Introdução

Este capítulo visa introduzir o aluno no processo de teste de software. Esta abordagem é uma da mais importantes no desenvolvimento de sistema. A sua operacionalização perfeita tem um impacto direto na organizações.

Detalhes da actividade

Teste é um conjunto de atividades que podem ser planificadas com antecedência e executadas sistematicamente. Por essa razão, deverá ser definido para o processo de software um modelo para o teste, um conjunto de etapas no qual pode-se colocar técnicas específicas de projecto de caso de teste e métodos de teste.

O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corregidas pela equipe de desenvolvimento antes da entrega final.

O teste visa aumentar aumentar a confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final.

Deve-se observar que há uma forte divergência de opinião sobre quais tipos de testes constituem "validação". Algumas pessoas acreditam que todo o teste é verificação e que a validação é executada quando os requisitos são examinados e aprovados, e mais tarde pelo usuário, com o sistema já em operação. Outras pessoas consideram o teste de unidade e de integração como verificação e o teste de ordem superior como validação.

Os testes de software normalmente são executados em diferentes estágios durante o ciclo de vida do desenvolvimento do software. Dependendo do objetivo principal do teste, quatro estágios são conhecidos:

Teste de unidade: descrito abaixo, o teste de unidade do âmbito e de escopo, realiza testes em componentes individuais de forma a determinar se cada um deles, separadamente, está sendo executado de maneira correta. Normalmente estes testes são de caixa branca, realizados pelos próprios desenvolvedores do componente.

Para a usa materialização os desenvolvedores de software usam ferramentas que provêm um suporte adicional para verificar a correctude do programa, como ferramenta de debugging ou framework para teste unitário, por exemplo. Os defeitos encontrados neste estágio são normalmente corrigidos de imediato, sem a necessidade de documentá-los formalmente, e assim, reduzindo o custo, pois antecipa a correção de defeitos. Geralmente é necessária a utilização de stubs (módulos que substituem outros módulos subordinados) e drivers (um módulo que substitui outro módulo que seja responsável por controlar a chamada de um sistema), para serem utilizados no lugar dos softwares que estejam eventualmente faltando e para simular a interface entre os componentes de software(Guerra & Colombo 2008).

Teste de integração ou teste de programas: Também do âmbito escopo, este tem como objectivo de integrar os testes feitas no teste de unidade em um só, bem como as interfaces entre os componentes. A integração deve ser realizada adicionando-se os componentes um por um, e após cada passo um teste é necessário (teste incremental).

"Esta técnica tem a vantagem de adiantar a detecção de defeitos no processo de testes e corrigi-los mais rapidamente, enquanto é mais fácil determinar as causas dos erros. Por outro lado, tem a desvantagem de ser uma prática bastante custosa. Sendo assim, a integração pode ser feita basicamente de duas formas: Top-down ou Bottom-up. Na primeira, os testes são realizados de cima para baixo (começando da GUI ou do menu principal); componentes ou sistemas são substituídos por stubs. Na segunda, os testes começam na parte mais básica do sistema até o nível mais alto; componentes ou sistemas são substituídos por drivers"(Guerra & Colombo 2008).

Teste de sistema: Visa assegurar que as funções do software em conformidade com as expectativas do cliente são razoáveis, tal forma como foi definido na secção critérios de validação para a Especificação dos Requisitos.

Neste estágio o propósito do teste está em verificar o funcionamento de todo o sistema, já integrado, e analisar se ele está de acordo com os requisitos que foram especificados. Neste momento, não só são realizados os testes de integração dos componentes do software entre si, como também destes componentes com um ambiente de teste correspondente à produção final (hardware, software, outros sistemas), de modo a minimizar o risco de que falhas relacionadas com o ambiente operacional do produto não sejam encontradas. Geralmente a estratégia de caixa preta é utilizada neste estágio, mas testes de caixa branca também podem ser realizados(Guerra & Colombo 2008).

Teste de aceitação: o teste de aceitação corresponde ao teste realizado pelo usuário de fato do sistema, no momento em que todos ou quase todos os defeitos encontrados nas etapas anteriores já tenham sido corrigidos. O propósito deste teste é estabelecer a confiança do sistema; ele está mais relacionado com a validação do sistema, em que está se tentando determinar se o sistema está de acordo com os requisitos especificados. Normalmente os testes de aceitação podem ser de duas categorias: testes alfa e testes beta. Os primeiros são realizados nas instalações do desenvolvedor, que fica observando os usuários utilizarem o sistema, e anotam os problemas identificados. Já os testes beta são realizados no ambiente real de trabalho do usuário, que instala o sistema e testa, sem a presença do desenvolvedor. Em seguida, um documento contendo os registros dos problemas encontrados é enviado à organização desenvolvedora(Guerra & Colombo 2008).

Tipos de testes - Quadro resumo de tipo de testes e sua aplicação

Tipo de teste		Função
Objectivo	Teste defeitos	● Tem por objetivo encontrar defeitos ; Normalmente realizados com protótipos funcionais
	Testes de Validação	● Utilizado para demonstrar ao desenvolvedor e ao cliente do sistema que o software atende aos seus requisitos. Um teste bem sucedido visa mostrar que o sistema opera conforme especificado pelo cliente.
Método	Testes Controlados	● Realizados em laboratório ou em condições operacionais sob a supervisão de um testador Baseados na geração de casos de testes – Podem ser realizados com protótipos funcionais
	Testes Estatísticos	● Utilizados para verificar como o software nas condições operacionais (versão beta) Avalia desempenho, robustez, confiabilidade, etc Utiliza programa de monitoramento e “logs” com ocorrências operacionais. – Exemplos de medições: • Número de falhas observadas • Tempos de resposta • Tempos de execução.
Escopo	Testes de Unidade	● Conhecidos como testes em ponto pequeno São testadas as unidades individuais funções, objetos, componentes.
	Testes de Integração	● Conhecidos como testes em ponto grande Os componentes são integrados e o conjunto maior é testado – módulo e sub-sistemas.

Tabela 1: Tipos de teste

Testes de integração incremental

Este tipo actua na posição horizontal e vai crescendo segundo as etapas.

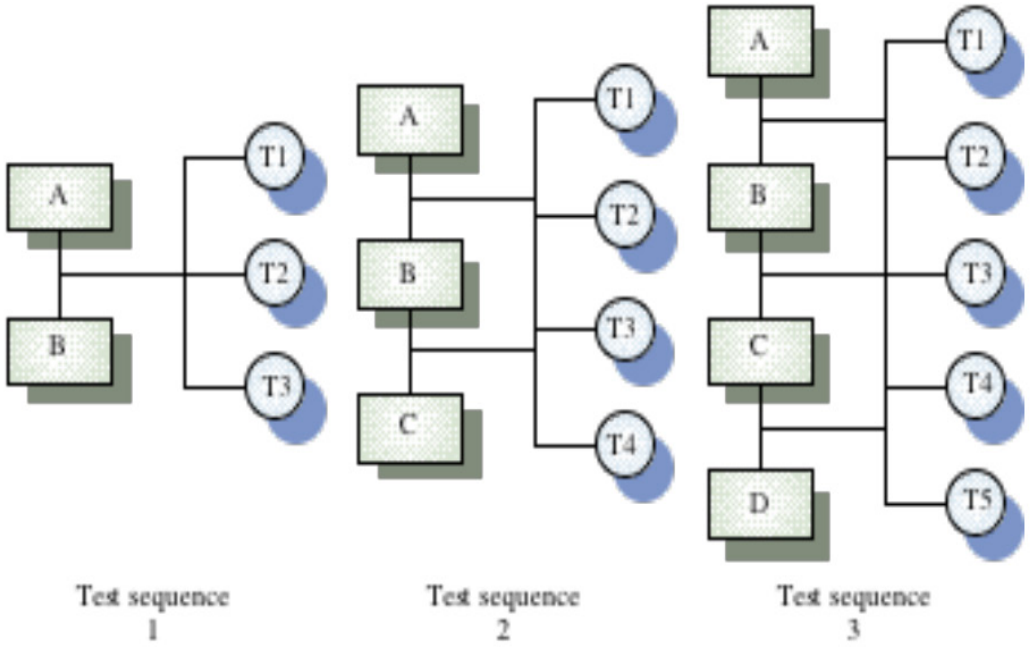


Figura: Testes de integração incremental

Fonte: (internet)

Testes de integração Top-down

Começa pelos componentes de alto nível e vai descendo na hierarquia de componentes, de acordo com a arquitetura do software. • Os subcomponentes são representados por stubs. •Stubs tem a mesma interface, mas não precisa ter funcionalidade. Basta retornar valores esperados.

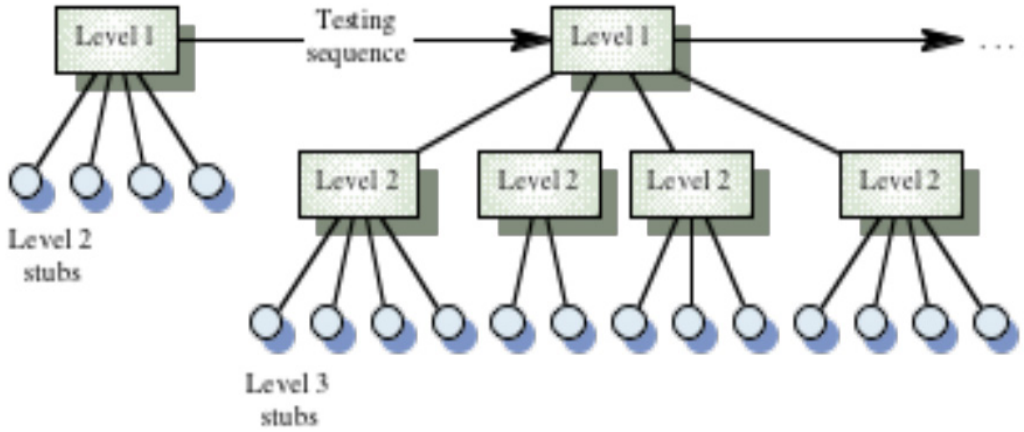


Figura: Testes de integração Top-down

Fonte: (internet)

Testes de integração Bottom-up

Integra unidades já testadas em módulos de mais alto nível, de acordo com a arquitetura.

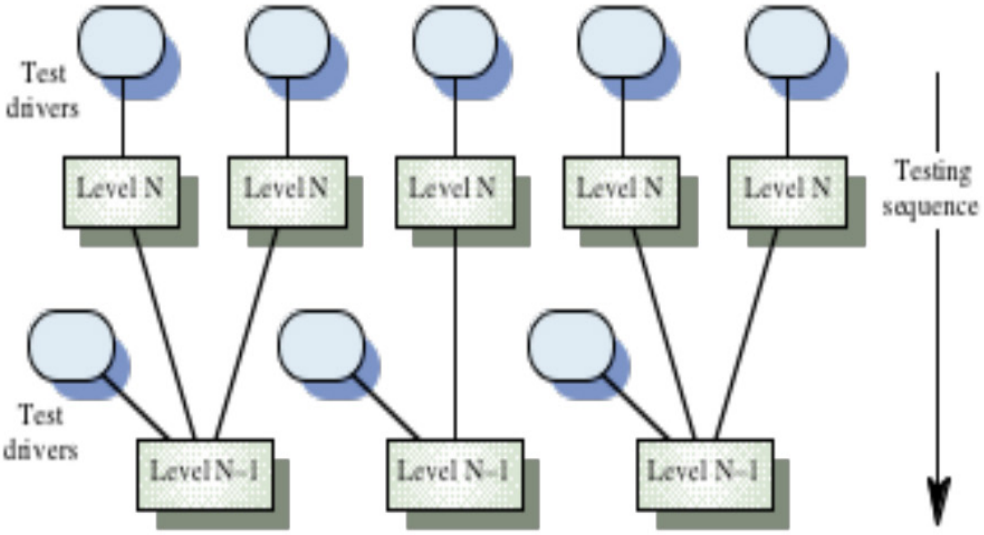


Figura: Testes de integração Button-down

Fonte: (internet)

Processo de testes

O processo de testes pode ser dividido basicamente em cinco etapas: planejamento e controle, análise e projeto, implementação e execução, avaliação de critério de saída e reportagem e atividades de encerramento de testes (Graham et. al 2007). Estas atividades são logicamente sequenciais, porém, em um projeto específico, podem se sobrepor, serem executadas em paralelo ou até mesmo serem repetidas.

Conclusão

Neste capítulo discutiu-se as a missão do teste como ferramenta crucial para a validação do produto. Debruçou-se ainda sobre diferentes tipos de testes e as principais abordagens.

Avaliação

1. O que entende por Verificação e Validação e diga qual dos dois termos ajuda o desenvolvedor a pautar por um software amigável?
2. Fale de diferentes abordagens de tipos de testes.
3. Qual é importância do teste?

Actividade 1.2 - Estratégias de teste

Introdução

No desenvolvimento de software existem várias etapas, desde o primeiro contacto entre as partes interessadas até o produto final. O impâcto positivo que o software possa verificar nas organizações tem a ver com a engenharia de requisitos. A análise de cada detalhe guia ao desenvolvedor a criar um produto que satisfaz ao usuário. Neste processo, uma das fases mais importantes é a verificação e validação do produto.

Neste actividade, iremos discutir diferentes estratégias de teste.

Detalhes da atividade

Uma estratégia de testes é uma abordagem geral para o processo de teste em vez de um método de concepção especial ou do sistema ou componente de testes. Podem ser adoptadas estratégias diferentes dependendo do sistema a ser testado e o processo de desenvolvimento utilizado.

Uma estratégia para o teste de um projecto descreve a abordagem geral e os objetivos das atividades de teste. A estratégia define as ferramentas e técnicas de teste a serem empregadas. A estratégia definida para testes comporta os seguintes passos implementados sequencialmente e pode ser vista sobre duas abordagens: a abordagem top-down e a button-up.

Abordagem top-down

Como o nome sugere, esta começa com o mais alto módulo de programa, módulos integrados e vai se movendo para baixo através da estrutura do programa. Pode ser feito em qualquer amplitude. esta abordagem top-down verifica os principais pontos de controlo e de decisão no início do processo de teste e uma vez que o módulo stubs substitua o código de baixo nível durante o teste, faz com que não haja significante fluxo de dado para cima e para dentro a estrutura do programa.

Abordagem top-down

Inicia a construção e testando com módulos no nível mais baixo na estrutura do programa.

A necessidade de stubs é removido - em vez de controladores são escritos para coordenar a entrada e saída de dados de teste. a abordagem bottom-up elimina a necessidade de programar stubs, o programa como uma entidade não existe até o último módulo é adicionado.

Existe uma terceira abordagem (estrategia) abordagem Thread ou Thread testing que foi concebido para sistemas de tempo real. É uma abordagem baseada em eventos onde os testes são baseados em eventos que acionam acções do sistema. esta abordagem é uma combinação de abordagem top-down e bottom-up e é frequentemente utilizado - testes sandwich. Este simplesmente adopta uma abordagem top-down para níveis superiores da estrutura do programa e uma abordagem bottom-up subalterno para módulos.

Testes de módulos atômicos: cada módulo (procedimentos ou funções) que não invoca outros módulos é testado individualmente, faz muito uso das técnicas de Teste de Caixa Branca (White Box), exercitando caminhos específicos da estrutura de controle de um módulo, a fim de garantir uma completa cobertura e máxima detecção de erros. no teste de Caixa Branca o testador está interessado no que está acontecendo dentro da caixa. É caracterizada por avaliar as funcionalidades internas dos componentes do software, baseando-se no código fonte e procurando exercitar estruturas de controle e de dados do programa. Sendo assim, faz-se necessário que o analista de testes tenha boa habilidade em programação de modo a entender todos os caminhos lógicos possíveis.

Testes de Integração: consiste na montagem e integração dos módulos básicos a fim de formarem um pacote de software.

Testes de Validação: consiste na realização de testes funcionais, baseados na especificação de requisitos funcionais, comportamentais e de desempenho. Os Critérios de Validação definidos após a fase de Análise de Requisitos devem ser testados. Utilizam-se exclusivamente Testes de Caixa Preta (Black Box). No teste de Caixa Preta o testador visualiza o software como uma caixa preta, ou seja, não considera a estrutura interna do programa, de que forma o código foi implementado ou que tecnologia foi utilizada, por exemplo. Considerando os dados de entrada, o objetivo principal é observar as saídas geradas pelo sistema e verificar se estas estão de acordo com o esperado

-Testes de Sistema: o software, uma vez validado, é combinado com outros elementos do sistema(por exemplo, hardware, pessoas, bancos de dados). Verificam se todos os elementos combinam-se adequadamente e se a função/desempenho global do sistema é conseguida.

Testes de Aceitação: consiste no processo de comparar o programa com seus requisitos iniciais e as necessidades dos usuários finais. É usualmente realizado pelo usuário final. Também denominado de Alfa -Teste (realizado nas dependências do desenvolvedor) ou Beta-teste (realizado nas dependências do usuário final).

Conclusão

Nesta actividade o foco foi nas estratégias que podem ser usadas para testar de formas diferentes o produto final ou em desenvolvimento. Discute-se a questão de white box, black box e thread.

Avaliação

1. O que são estratégias do teste?
2. Quais são as principais abordagens de teste de software?
3. Discuta as seguintes abordagens

    a. Black Box;

    b. White Box e;

Actividade 1.3 - Validação de Design preliminares através de prototipagem.

Introdução

O velho ditado "uma imagem vale por mil palavras" capta de prototipagem. Prototipagem utiliza recursos visuais para descrever milhares de palavras de design e desenvolvimento de especificações para uma aparência e comportamento do sistema. Em uma abordagem iteractivo para o design do sistema, prototipagem

A Prototipagem é um processo rápido de construção do um sistema de forma lúdica através de uso de diferentes ferramentas tais como CAD, seja ele um site ou aplicativo e validar com uma audiência mais ampla de usuários interessados, os desenvolvedores e designers. Fazer isso rapidamente e de maneira iterativa gera feedback cedo e frequentemente no processo, melhorando o design final e reduzindo a necessidade de mudanças durante o desenvolvimento.

Protótipos de papel áspero esboços para simulações interactivas que se parecem e funcionam como produto final. A chave para o êxito de protótipos estão a rever rapidamente com base no feedback e usando a abordagem de prototipagem adequado. Prototipagem ajuda os desenvolvedores a experimento com múltiplas abordagens e ideias. Facilita o debate através de visuais em vez de palavras, ele garante que toda a gente partilha um entendimento comum e que reduz o risco e evita os requisitos perdidas, levando a uma melhor concepção rapidamente.

Detalhes da atividade

Prototipação envolve várias iterações de um processo de três etapas:

- Protótipo - Converter a descrição dos utilizadores da solução em um acúmulo, factoring na experiência do usuário normas e melhores práticas.
- Revisão - Compartilhar o protótipo com usuários e avaliar se ela atende às suas necessidades e expectativas.
- Refinar - baseadas no feedback, identificar áreas que precisam ser refinados ou mais definidos e esclarecidos.

O protótipo geralmente começa pequeno, com algumas áreas chave edificado, e cresce em amplitude e profundidade ao longo de várias iterações como áreas requeridas são construídos para fora, até que o protótipo seja finalizado e entregue para o desenvolvimento do produto final. A rapidez do processo é mais evidente em iterações, que vão desde as mudanças em tempo real para ciclos de iteração de poucos dias, dependendo do escopo do protótipo.

O escopo de um protótipo

A palavra protótipo capta imagens de uma frequentemente codificados e totalmente funcional versão de um aplicativo ou interface. Protótipos não são destinados a evoluir para soluções totalmente funcionais, mas se destinam a ajudar os utilizadores a visualizar o artesanato e

a experiência do usuário do produto final. Com isso em mente, quando o escopo de um protótipo, decidir sobre algumas questões fundamentais antes de iniciar qualquer trabalho de prototipagem.

Bons candidatos para prototipagem incluem interações complexas, nova funcionalidade e as alterações no fluxo, tecnologia ou design. Por exemplo, prototipagem resultados da pesquisa é útil quando você deseja alterar significativamente a experiência de pesquisa padrão; dizer, introduzir pesquisa facetada ou a capacidade de visualizar um documento sem deixar os resultados da pesquisa.

Uma boa regra é focar a 20% da funcionalidade que será usado 80% do tempo; ou seja, a funcionalidade de tecla que será usada com mais frequência. Lembre-se de que o ponto de prototipagem é a demonstração de como algo vai trabalhar ou, em fases posteriores, o que o projeto será a aparência, prototipagem sem o produto inteiro.

Depois de identificar as áreas a que foram protótipos, mesclar juntos em um ou mais cenários: identificar os caminhos coerente através da experiência do usuário que o protótipo simula. Por exemplo, um site que vende calçado, um cenário poderia ser "Chata Joe" comprar exatamente o mesmo Nike tênis de corrida que ele comprou há seis meses, enquanto outro cenário poderia ser "Explorar Sam" navegando por tamanho 10s para encontrar um par de Oxfords e par de aos vadios que lhe interesse.

Planifique a sua iterações

Todo o protótipo geralmente não é construído em uma única iteração mas pedaço a pedaço. Uma boa abordagem é começar a prototipagem amplamente e amplamente e depois coloque o foco em áreas selecionadas da solução. Para um site, isso significaria construir a "página inicial" e desembarque de páginas para as seções principais na primeira iteração (por vezes referido como um protótipo horizontal) e depois analisar e rever esse quadro. As iterações subsequentes poderiam diminuir em uma ou mais secções do site (um protótipo vertical); para um site de download de mídia, este poderia ser os passos que um usuário iria tomar para encontrar um vídeo e baixá-lo, ou a forma como eles seria gerir o material em sua biblioteca on-line.

Escolha o tipo apropriado de fidelidade

A fidelidade se refere à forma como a ilusão de um protótipo se assemelha a solução final. Existem múltiplas dimensões da fidelidade, e protótipos podem mentir em qualquer lugar sobre o espectro para cada uma destas dimensões. Figura abaixo, demonstram o nível de fidelidade diferentes.

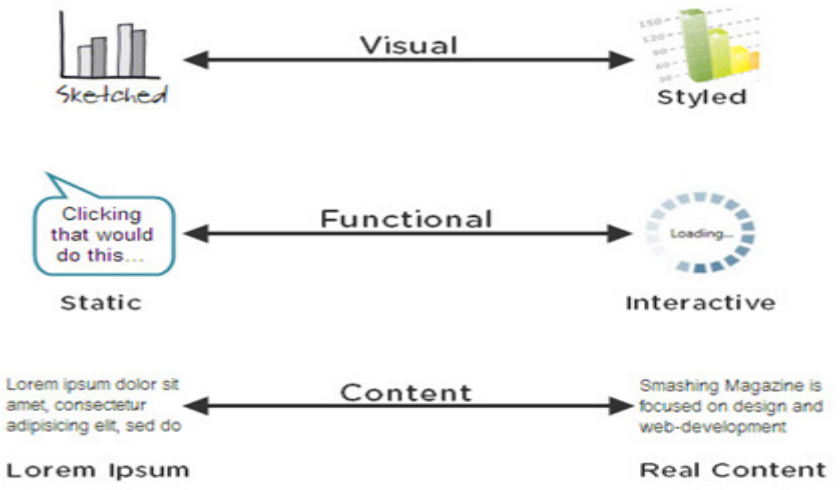


Figura : Níveis de fidelidade

Fonte: (internet)

Dependendo do estágio do processo de design e os objectivos do protótipo, selecione a fidelidade como mostrado na figura 1 para cada uma das seguintes dimensões:

Fidelidade visual (esboçado ↔ estilo)

Olhar e sentir são a dimensão mais visível de um protótipo de sua fidelidade e, se não for adequadamente selecionada, pode contornar o protótipo de clientes. Por exemplo, vai "hi-fi" demasiado cedo e usuários incidirá sobre o design visual, o que não é apropriado nas fases precoces. A partir de um ponto de vista visual, protótipos não têm de ser perfeitas mas deve ser proporcional. Por exemplo, se a área de navegação esquerda tem de ocupar um quinto de um 1024-pixel tela, ele não precisa ser exatamente 1024 pixels de largura, enquanto é proporcionalmente representadas no protótipo. Como o desenvolvimento de protótipos progride através do ciclo de design aumentar a fidelidade visual conforme necessário introduzindo elementos de estilo, cor, brinding e gráficos.

Fidelidade funcional (estático ↔ interativa)

O protótipo revelam o modo como a solução funcionará (estático) ou não parece ser totalmente funcional e responder à entrada do usuário (Interativo)? Esta dimensão é menos uma distracção para os usuários, mas adicionar interatividade em subsequentes iterações aumenta a fidelidade funcional e permite que o protótipo a ser utilizado para testes de usabilidade e formação e comunicações.·

A fidelidade dos conteúdos (Lorem ipsum ↔ conteúdo real)

Outra dimensão que muitas vezes distrai os usuários é o conteúdo que é exibido no protótipo. Linhas de torção e Detonador inerte de texto como Lorem ipsum são úteis para evitar nas fases precoces da prototipagem. Mas como o protótipo é refinado, avaliar a necessidade de substituir o detonador inerte texto com conteúdo real para sentir como ele afeta o projeto geral.

O espectro de Prototipagem

Baixa Fidelidade

A maneira mais rápida para iniciar também é a maneira mais fácil de prototipagem: colocando a caneta/lápis para papel. O esboço em papel é uma abordagem de baixa fidelidade que qualquer pessoa pode fazer; sem ferramentas especiais ou experiência necessário. Utilizado com mais frequência durante as primeiras fases de um ciclo de design esboçar é uma forma rápida de criar compilações irregular de abordagens de design e conceitos e obter feedback dos usuários. Prototipagem de papel é ideal durante o debate e a conceituação e só pode ser feito em uma área de trabalho com um caderno de rascunhos ou em um grupo com um flip-chart ou quadro de comunicações e marcadores. A figura 2 apresenta o papel esboços para ilustrar a baixa fidelidade.

Situada na extremidade de baixa fidelidade do espectro de prototipagem, protótipos de papel são estáticos e normalmente têm baixo conteúdo visual e de fidelidade. Esta força os usuários a se concentrar em como vão usar o sistema em vez de o que ela terá a aparência e torna mais aberto para alterações designers com base no feedback dos usuários. Prototipagem de baixa fidelidade é mais fácil de aprender e permite fazer alterações de forma fácil e rápida.

Médio Fidelidade

Como começar a usar ferramentas baseadas em computador como o Visio e Omnigraffle ao protótipo, a fidelidade aumenta na maioria das frentes, produzindo média fidelidade protótipos. Wireframes, fluxos de tarefas e cenários que são criados com estas ferramentas levam mais tempo e esforço mas olhar mais formal e refinado. Enquanto elementos visuais de marca, cores e estilo pode ser introduzido, prototypers muitas vezes ficar longe deles, concentrando-se em vez de demonstrar o comportamento do aplicativo.

A interactividade pode ser simulado por meio de vinculação de páginas ou telas, mas funcional fidelidade aqui é médio no melhor. Estes protótipos são mais adequados para determinar se as necessidades do usuário são cumpridos e se a experiência do usuário é ideal. Uma captura de tela na Figura 3 ilustra um protótipo de média fidelidade.

Há duas razões pelas quais um pode intencionalmente fazer uma média fidelidade protótipo não ter a aparência de uma média fidelidade protótipo:

A primeira é que, usando Balsamiq ou esparsas stencils visio para tornar a aparência do protótipo de baixa fidelidade, você forçar os utilizadores a visualizar a ti como um projecto ou trabalhos em curso, em vez de um polido e produto acabado.

· A segunda é que, dando o protótipo de uma alta fidelidade visual (por exemplo, em um layout abrangente feito no Photoshop), você obtém o usuário a se concentrar sobre o design visual e a aparência e sensação, incluindo cores, fontes, layout, logotipo e imagens.

A velocidade de média fidelidade prototipagem é alcançada com modelos, stencils e reutilizáveis widgets e elementos. Ele recebe mais rápido à medida que você se torna mais proficiente com suas ferramentas de escolha.

Alta Fidelidade

Protótipos de alta fidelidade são os mais realistas e muitas vezes são confundidos com o produto final, mas eles são geralmente consomem muito tempo. Uma captura de tela na Figura 4 ilustra um protótipo de alta fidelidade.

Há alguns anos, a única forma de criar protótipos de alta fidelidade foi efetivamente código usando uma linguagem de programação, que muitas vezes necessário o designer e desenvolvedor para trabalhar juntos. Estes dias, todavia, a aplicação de ferramentas de simulação permitem que usuários não técnicos para arrastar e soltar widgets de interface para criar protótipos de alta fidelidade que simulam a funcionalidade do produto final, mesmo para a lógica de negócios e interações de banco de dados.

Axure e iRise são alguns exemplos de aplicação de ferramentas de simulação que pode ser usado para criar protótipos de alta fidelidade.

Estes protótipos são adequadas quando a alta fidelidade visual e funcional é necessário; por exemplo, quando a introdução de uma nova tecnologia (dizer, quando se deslocam de um aplicativo de mainframe (sim, ainda existem!) para uma solução baseada na Web. A maioria destes protótipos não podem ser convertidos em código utilizável, mas eles servem como uma excelente referência para desenvolvedores. Estes também são úteis para a realização de testes de usabilidade e treinamento de usuários.

Prototipagem de alta fidelidade é relativamente rápida, considerando o nível de interactividade e fidelidade envolvidos, e ele pode ser acelerado pelo usando arrastar e soltar ferramentas de simulação. Além disso, algumas dessas ferramentas facilitar a recolha de feedback do usuário e documentação de requisitos, acelerando ainda mais o processo de design. Mesmo que você não precisa aprender uma nova linguagem de programação, essas ferramentas têm uma curva de aprendizado.

A seleção de um nível de fidelidade

Na escolha do protótipo fidelidade, não é uma abordagem correcta. A maioria dos projetos de novos produtos são melhores começou com esboços, então movendo a médio ou protótipos de alta fidelidade, dependendo da complexidade do sistema e os requisitos das dimensões da fidelidade.

Exemplos: Em uma indústria farmacêutica projeto de desenvolvimento de aplicativo, começando a partir de quadros brancos interativos para protótipos que têm alto conteúdo e funcionais fidelidade mas baixa fidelidade visual, o cliente pode se preocupar menos com a aparência do que sobre as diretrizes corporativas aderindo a.

Em um projeto de desenvolvimento de aplicativos de varejo, um protótipo interativo pode precisar de alta fidelidade visual e funcional. A fidelidade dos conteúdos não importa porque os clientes seria a reutilização de conteúdo e já estavam familiarizados com ele. Para eles, a aparência e experiência interactiva importava mais porque esta foi a sua primeira implementação da aplicação.

Ferramentas de seleção

Dependendo da sua abordagem, você tem uma grande variedade de ferramentas para escolher. Cada ferramenta tem seu próprio conjunto de recursos e pontos fortes. Com base nas suas necessidades e os requisitos dos projectos você trabalhar, avaliar qual ferramenta seria mais adequado. Aqui estão algumas questões a considerar ao avaliar ferramentas:

- Como é fácil de aprender e usar a ferramenta?
- É flexível para apoio dos protótipos para Web, embalados e aplicativos de software personalizados, bem como desktop e aplicativos móveis?
- Existe um repositório de modelos reutilizáveis stencils, ou widgets disponíveis?
- Quão fácil é para compartilhar o protótipo com outros para revisão? Pode o seu feedback ser capturado com a ferramenta?
- Quão fácil é para fazer alterações ou para incorporar o feedback?
- Não tem recursos de colaboração, tais como permitir que várias pessoas a trabalhar sobre ela ao mesmo tempo?
- Quais são os termos de licenciamento e os custos?

Conclusão

Nesta actividade foram discutidas questões relacionadas com uma das mais promissoras técnicas de modelação e desenho de software. A prototipação é um dos métodos que de forma mais interactiva ajuda aos analistas e desenvolvedores dos sistemas a executarem as suas tarefa de uma forma simplificada.

Avaliação

1. O que é a prototipagem?
2. Qual é o escopo de um protótipo

Resumo da Unidade

Nesta unidade foram discutidos temas importante para um analista e desenvolvedor de sistemas com objectivo de dotar o aluno de competência na área de engenharia de sistema. Tratou e do conceito verificação e validação com destaque no seguintes elemento: - missão de testes, estratégias de teste, técnicas de testes de conformidade e desenhos Preliminares de validação através de prototipagem.

Avaliação da Unidade

Verifique a sua compreensão!

Teste de unidade 3

Instruções

Leia a unidade na sua totalidade e responda as questões que se seguem.

As seis (6) Questões valem 20 pontos. A 1 questão equivale a 5 pontos e as restantes 3 pontos cada.

Avaliação

1. Verificação e Validação (V&V) é um processo de verificação e análise contínua, onde as actividades decorrem em cada fase do processo de software, ele abrange toda as etapas de produção iniciando com as análise de requisitos até os testes de produtos.

    a. Diferencie os dois processos.

    b. Fale de diferentes testes e enquadre-os a cada um dos processos.

2. O que entendes por estratégias de teste?
3. Em que consiste o escopo de um protótipo?
4. Quais são os critérios para a escolha de um tipo apropriado de fidelidade?
5. Diferencie cada uma das fidelidades dos protótipos?
6. Discuta sobre as ferramentas de seleção.

RESPOSTAS DA UNIDADE

UNIT 3 - Actividade 1.1

1. Verificação e Validação de software é o acto de efetuar testes em tudo que construímos e usamos é algo natural do ser humano, isso porque temos um instinto natural de saber se as coisas funcionam corretamente. Testar tem se mostrado ao longo dos séculos a maneira mais eficiente de verificar se algo está se comportando de forma correta. Produtos de software não fogem a esta regra. Isso porque mesmo o sistema que foi minunciosamente projetado e construído não está livre de apresentar falhas, defeitos e erros. Para que estes problemas sejam descobertos antes do sistema ser entregue para o cliente, executamos uma série de atividades que chamamos de Validação, Verificação.

A actividade de verificação resume-se em responder a esta pergunta. A verificação tem o objetivo de avaliar se o que foi planejado realmente foi realizado. Ou seja, se os requisitos e funcionalidades documentados foram implementados, além disso a verificação também pode ser realizada para especificação de sistemas, para avaliar se os requisitos estão sendo documentados como deveriam e ainda prever falhas ou inconsistências entre requisitos e a validação tem o objetivo de avaliar se o que foi entregue atende as expectativas do cliente.

Ou seja, se os requisitos, independente do que foi planejado, estão sendo implementados para atender a regra de negócio do cliente, se o sistema é realmente aquilo que o cliente quer e está pagando para ter. A validação final do sistema é realizada pelo próprio cliente ou usuário.

2. O quadro abaixo resume os tipos de testes.

Tipo de teste		**Função**
Objectivo	Teste defeitos	· Tem por objetivo encontrar defeitos ; Normalmente realizados com protótipos funcionais
	Testes de Validação	· Utilizado para demonstrar ao desenvolvedor e ao cliente do sistema que o software atende aos seus requisitos. Um teste bem sucedido visa mostrar que o sistema opera conforme especificado pelo cliente.
Método	Testes Controlados	· Realizados em laboratório ou em condições operacionais sob a supervisão de um testador Baseados na geração de casos de testes – Podem ser realizados com protótipos funcionais
	Testes Estatísticos	· Utilizados para verificar como o software nas condições operacionais (versão beta) Avalia desempenho, robustez, confiabilidade, etc Utiliza programa de monitoramento e "logs" com ocorrências operacionais. – Exemplos de medições: • Número de falhas observadas • Tempos de resposta • Tempos de execução.
Escopo	Testes de Unidade	· Conhecidos como testes em ponto pequeno São testadas as unidades individuais funções, objetos, componentes.
	Testes de Integração	· Conhecidos como testes em ponto grande Os componentes são integrados e o conjunto maior é testado – módulo e sub-sistemas.

3. Teste de software é a etapa de controle de qualidade, serve para assegurar que o software está contemplando todas as funcionalidades esperadas e que estas estão funcionando corretamente. Geralmente é a última etapa na construção de um sistema, aplicativo ou game a ser lançado, podendo ser um novo projeto ou uma nova versão. Entre as tarefas de um profissional de teste, as principais são: entendimento do projeto e novas funcionalidades, planejamento dos testes que serão executados, execução dos testes e registro de defeitos. O profissional de testes deve integrar de um time de desenvolvimento e estar a par do projeto como um todo, pois além de validar as novas funcionalidades, também deve validar o layout, usabilidade e performance. Para otimizar seu trabalho, o testador ou equipe de testes, pode utilizar ferramentas de automatização. Com isso, há um ganho de tempo muito grande, pois a ferramenta pode realizar testes pré-gravados de forma automática e mais rápida. Embora algumas empresas e equipes ainda desenvolvam softwares sem a participação de um testador ou analista de testes, esse comportamento tende a ser cada vez mais raro, pois ter um profissional focado na qualidade do software é indispensável e pode comprometer o produto e a imagem da empresa.

UNIT 3 - Actividade 1.2

1. Uma estratégia de testes é uma abordagem geral para o processo de teste em vez de um método de concepção especial ou do sistema ou componente de testes. Podem ser adoptadas estratégias diferentes dependendo do sistema a ser testado e o processo de desenvolvimento utilizado

2. . Existem duas abordagens a referir: Top-Down e a Button-UP. A abordagem top-down como o nome sugere, esta começa com o mais alto módulo de programa, módulos integrados e vai se movendo para baixo através da estrutura do programa e a button up Inicia a construção e testando com módulos no nível mais baixo na estrutura do programa.

3. Resposta

a) Teste de caixa branca

O analista tem acesso ao código fonte, conhece a estrutura interna do produto sendo analisado e possibilita que sejam escolhidas partes específicas de um componente para serem avaliadas. Esse tipo de teste, também conhecido como teste estrutural, é projetado em função da estrutura do componente e permite uma averiguação mais precisa do comportamento dessa estrutura. Perceba que o acesso ao código facilita o isolamento de uma função ou ação, o que ajuda na análise comportamental das mesmas.

b) Teste de caixa preta

O analista não tem acesso ao código fonte e desconhece a estrutura interna do sistema. É também conhecido como teste funcional, pois é baseado nos requisitos funcionais do software. O foco, nesse caso, é nos requisitos da aplicação, ou seja, nas ações que ela deve desempenhar. Para mostrar quais problemas que esse tipo de teste rastreia. Enfim, todo tipo de falha funcional, ou seja, falhas que contrariam os requisitos da aplicação.

Há que se destacar, contudo, que existe um elemento comum aos dois tipos de teste. Tanto no teste de caixa branca quanto no teste de caixa preta, o analista não sabe qual será o comportamento da aplicação ou do alvo de teste em uma determinada situação. A imprevisibilidade de resultados é comum aos dois casos.

UNIT 3 Actividade 1.3

1. Prototipagem ajuda os desenvolvedores a experimento com múltiplas abordagens e ideias. Facilita o debate através de visuais em vez de palavras, ele garante que toda a gente partilha um entendimento comum e que reduz o risco e evita os requisitos perdidas, levando a uma melhor concepção rapidamente.

2. A palavra protótipo capta imagens de uma frequentemente codificados e totalmente funcional versão de um aplicativo ou interface. Protótipos não são destinados a evoluir para soluções totalmente funcionais, mas se destinam a ajudar os utilizadores a visualizar o artesanato e a experiência do usuário do produto final. Com isso em mente, quando o escopo de um protótipo, decidir sobre algumas questões fundamentais antes de iniciar qualquer trabalho de prototipagem.

Leituras e outros Recursos

- Belgamo, A. & Martins, L.E.G., 2000. Estudo Comparativo sobre as técnicas de Elicitação de Requisitos do Software. In XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná.
- Guerra, A.C. & Colombo, R.M.T., 2008. Qualidade de Produto de Software. PBQP/MCT. Disponıvel em:< http://www. mct. gov. br/index. php/content/view/2867. html# lista>. Acesso em, 13(09), p.2009.
- Jalote, P., 2008. A Concise Introduction to Software Engineering Sétima Edição. Springer, ed.,
- Nazareth, D., 1999. Assembling a Metrics Suite for Rule-Based Systems Development. AMCIS 1999 Proceedings, p.23.
- Pressman, R.S., 2011. Engenharia de Software - Uma Abordagem Profissional Sétima Edição. M. G. Hill, ed.,
- Rocha Balthazar, G. da, 1981. Visão Geral da Qualidade de Software. Revista Eletrônica da Faculdade Metodista Granbery-http://re. granbery. edu. br-ISSN, p.0377.
- Santos Bueno, C. de F. dos & Campelo, G.B., Qualidade de Software.
- Gerard O'Regan, Introduction to Software Quality, Springer, 2014
- Jeff Tian, Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement , IEEE Computer Society, 2005
- Darrel Ince, An Introduction to Software Quality Assurance and Its Implementation, McGraw-Hill, 1994
- https://en.wikipedia.org/wiki/Problem_frames_approach[Accessed 23/02/2016]
- Alistair Sutcliffe, User-Centred Requirements Engineering, Springer-Verlag London Limited 2002

- https://en.wikipedia.org/wiki/Requirements_analysis [Accessed 23/02/2016]
- David J. Gilmore, Russel L. Winder and Francoise Detienne, User-Centred Requirements for Software Engineering Environments, Springer-Verlag Berlin Heidelberg GmbH, 1994
- http://www.rbcs-us.com/documents/Defining-Testing-Article.pdf [accessed 24/02/2016]
- http://www.sqa.org.uk/e-learning/SDPL03CD/page_16.htm[accessed 24/02/206]
- http://searchsoftwarequality.techtarget.com/definition/conformance-testing[accessed [24/02/2016]
- Paul Ammann and Jeff Offutt, INTRODUCTION TO SOFTWARE TESTING, CAMBRIDGE UNIVERSITY PRESS, 2008
- William E. Perry, Effective Methods for Software Testing, Third Edition, Wiley Publishing, Inc., Indianapolis, Indiana, 2006
- http://www.softwaretestinghelp.com/what-is-conformance-testing/[accessed 24/02/2016]
- https://www.smashingmagazine.com/2010/06/design-better-faster-with-rapid-prototyping/[accessed 24/02/2016

Unidade 4. Gestão da Qualidade

Introdução à Unidade

Inúmeras são organizaçoes que estão buscando melhorias em seus processos de desenvolvimento de software. A melhoria não consiste apenas no software em si, mas envolve o relacionamento com o cliente também, por meio de planeamento e redução do número de defeitos nos produtos vendidos.

As principais referências para melhoria de processos utilizados pelas organizações que desenvolvem ou adquirirem produtos de software são:

CMM – Capability Maturity Model é um modelo desenvolvido pelo Software Engineering Institute (SEI) para melhoria da maturidade dos processos de desenvolvimento de software.

ISO 9001 – Sistemas de qualidade; é um modelo para garantir a qualidade em projetos, desenvolvimento, instalação e serviços associados. São normas utilizadas como referência por muitas empresas desenvolvedores de software no mundo inteiro.

ISO 12207 – Tecnologia da Informação/Processo de ciclo de vida de software; esta norma estabelece uma estrutura comum para os processos de ciclo de visa de um software, cobrindo desde a concepção até a retirada do software do mercado.

ISO 15504 (SPICE) – Tecnologia da Informação/Avaliação de processos; norma desenvolvida pela ISO IEC em conjunto com o projeto SPICE para avaliação de processos.

Ainda existem outras referências, tais como:

- CMMI – Capability Maturity Model Integration; também desenvolvido pela SEI (Software Engineering Institute) visando a melhora da maturidade dos processos em uma organização. Foi desenvolvida a partir de três modelos:
- SW-CMM (Capability Maturity Model for Software);
- EIA/IS (Eletronic Interim Alliance Interim Standard)
- IPD-CMM (Integrated Product Development Capability Maturity Model).
- PMBoK – Project Management Body of Knowledge; conjunto de Conhecimento em Gerência de Projetos.

Objetivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Descrever o conceito de gestão de qualidade
- Desenvolver um quadro de normas e procedimentos organizacionais que conduzem a desenvolvimento de software de alta qualidade
- Selecionar os procedimentos adequados e normas e adaptar para um projecto especifico de software.
- Aplicar os procedimentos de qualidade e normas.

Termos-chave

Gestão da Qualidade: verificação independente sobre o software e desenvolvimento de software processo para promover a qualidade.

Normas de qualidade: Um quadro para a aplicação do processo de garantia de qualidade.

Plano de qualidade: Define os requisitos de qualidade de software e descreve como estes têm de ser avaliados.

Controle de qualidade: Acompanhamento do processo de desenvolvimento de software para garantir que procedimentos de garantia de qualidade e normas estão a ser seguido

Actividades de Aprendizagem

Actividade 1.1 - Garantia de qualidade e normas

Introdução

Garantia de qualidade é o processo de definição de como a qualidade do software podem ser alcançados e como a empresa de desenvolvimento sabe que o software tenha o nível de qualidade exigido. A actividade principal do processo de garantia de qualidade é a seleção e a definição de normas que são aplicadas para o processo de desenvolvimento de software ou produto de software.

Detalhes da actividade

Existem dois tipos principais de normas, o produto as normas e o processo de normas. Os padrões produto que são aplicados ao produto de software, ou seja saída do processo de software. Os padrões de processo definem os processos que devem ser seguidas durante o desenvolvimento de software. Os padrões do produto são baseados nas melhores práticas e fornecem um quadro para a aplicação do processo de garantia de qualidade.

O desenvolvimento de normas de projecto de engenharia de software é um processo difícil e demorado. Organizações Nacionais e Internacionais tais como ANSI e o IEEE desenvolveram padrões que podem ser aplicados a projectos de desenvolvimento de software. As normas de organização, desenvolvido por equipes de garantia de qualidade deve ser baseada sobre estas normas nacionais e internacionais. A tabela 1 mostra exemplos de normas de processo e de produto.

Tabela 1: Exemplos de normas de processo e de produto

Normas de produto	Normas de processo
Estrutura do documento de requisitos	Processo de aprovação do plano de projeto
Método do formato do cabeçalho	Processo de liberação de versão
Estilo de programação Java	O processo de controle de alterações
Formulário Solicitação de Mudança	Processo de gravação de teste

ISO

ISO 9000 é um conjunto de normas internacionais que podem ser usados no desenvolvimento de um sistema de gestão da qualidade em todas as indústrias. Normas ISO 9000 pode ser aplicada a uma variedade de organizações de fabricação para indústrias de serviços. ISO 9001 é o mais geral dessas normas. Ele pode ser aplicado a organizações que projetar, desenvolver e manter os produtos e desenvolver os seus próprios processos de qualidade. Um documento de apoio (ISO 9000-3) interpreta ISO 9001 para desenvolvimento de software.

A norma ISO 9001 não é específico para o desenvolvimento de software mas inclui princípios gerais que podem ser aplicados a projetos de desenvolvimento de software. A norma ISO 9001 descreve vários aspectos do processo de qualidade e define as normas de organização e procedimentos que uma empresa deve definir e seguir durante o desenvolvimento do produto. Estas normas e procedimentos são documentados em um manual de qualidade organizacional.

A norma ISO 9001 não define os processos de qualidade que deve ser usado no processo de desenvolvimento. As organizações podem desenvolver seus processos de qualidade e eles ainda podem ser compatíveis com ISO 9000 empresas. A norma ISO 9000 apenas requer a definição de processos para ser usado em uma empresa e ele não está preocupado com garantir que estes processos fornecem as melhores práticas e a elevada qualidade dos produtos. Por conseguinte, a certificação ISO 9000 não significa exactamente que a qualidade do software produzido pela ISO 9000 empresas certificadas será melhor do que o software de uma empresa não certificadas.

Normas de documentação

Normas de documentação em um projeto de software são importantes porque os documentos podem representar o software e o processo de software. Documentos padronizados têm uma aparência consistente, a estrutura e a qualidade, e deve por isso ser mais fácil de ler e entender. Existem três tipos de padrões de documentação:

Normas de processo de documentação - Essas normas definem o processo que deve ser seguido para a produção de documentos.

Padrões de documento - Estas normas descrevem a estrutura e apresentação de documentos.

Normas para o intercâmbio de documentos - Estas normas garantem que todas as cópias eletrônicas de documentos são compatíveis.

Conclusão

A actividade principal do processo de garantia de qualidade é a seleção e a definição de normas que são aplicadas para o processo de desenvolvimento de software ou produto de software. Esta atividade apresentou os dois tipos de normas aplicadas no processo de garantia de qualidade do software. As normas ISO e a documentação.

Avaliação

1. Quais são os dois tipos de normas
2. Desenhe um quadro resumo de normas de produto e de processo.

Actividade 1.2 - Planificação de Qualidade

Introdução

A planificação de qualidade é o processo de desenvolvimento de um plano de qualidade para um projecto. Neste capitulo discute-se o termo plano de qualidade de modo a ajudar ao aluno a aplicar esses conhecimentos na engenharia de software.

Detalhes da atividade

O plano da qualidade define os requisitos de qualidade de software e descreve a forma como estes devem ser avaliadas. O plano de qualidade seleciona os padrões organizacionais que são adequadas a um determinado produto e processo de desenvolvimento. Plano de qualidade tem as seguintes peças:

1. Introdução do produto.
2. Planos de produto.
3. Descrições de processos.
4. Metas de Qualidade.
5. Riscos e gestão do risco.

O plano da qualidade define os mais importantes atributos de qualidade para o software e inclui uma definição do processo de avaliação de qualidade. V A tabela 2 mostra os atributos de qualidade de software geralmente usados que podem ser considerados durante o processo de planejamento de qualidade.

Tabela 2: atributos de Qualidade do Software

Segurança	A compreensibilidade	Portabilidade dos números
Segurança	Testabilidade	Usabilidade
Confiabilidade	Adaptabilidade	Capacidade de reutilização
Resiliência	Modularidade	Eficiência energética

Robustez	Complexidade	Learnability
Manutenção		

Conclusão

Esta actividade descreveu o processo de planificação de qualidade.

Avaliação

O que é o plano de qualidade?

Identificar as principais partes de um plano de qualidade

Actividade 1.3: Controlo de qualidade

Introdução

O controle de qualidade assegura o acompanhamento do processo de desenvolvimento de software para garantir que os procedimentos de garantia de qualidade e normas estão sendo seguidas. Os resultados do processo de desenvolvimento de software são verificadas contra o projecto definido normas no processo de controlo de qualidade. A qualidade dos materiais de entrega do projeto de software pode ser verificado pela realização regular de qualidade de clientes e/ou avaliação automatizada de software. Qualidade de clientes são realizadas por um grupo de pessoas. Eles revisão do software e processo de software a fim de verificar que o projecto normas foram seguidas e que o software e os documentos em conformidade com essas normas. Os processos de avaliação automatizada de software o software por um programa que compara com as normas aplicadas para o desenvolvimento do projecto.

Detalhes da atividade

Avaliação de Qualidade

Avaliação de Qualidade é o método mais amplamente usado de validar a qualidade de um processo ou produto. Elas envolvem um grupo de pessoas a examinar uma parte ou a totalidade de um processo de software de sistema, ou sua documentação associada para descobrir potencias problemas. As conclusões da revisão são formalmente registradas e passadas para o autor para corrigir problemas descobertos. A tabela 3 descreve diversos tipos de revisão, incluindo análises de qualidade.

Tabela 3: Tipos de revisão

Tipo de revisão	Objectivo principal
Design ou programa de inspeções	Para detectar erros detalhados nos requisitos, design ou código.

Progresso de avaliação	Para fornecer informações para a gestão geral sobre o andamento do projeto.
Qualidade de avaliação	Para efetuar uma análise técnica dos componentes do produto ou da documentação para encontrar as incompatibilidades entre a especificação e o projeto de componentes, código ou documentação e para garantir que as normas de qualidade definidas da organização tenham sido seguidas.

Medição de Software e métricass

Medição de Software fornece um valor numérico para o atributo de qualidade de um produto de software ou a um processo de software. Comparação desses valores numéricos uns aos outros ou a normas tira conclusões sobre a qualidade dos processos de software ou software. Medições de produto de software pode ser usado para fazer previsões gerais sobre um sistema de software e identificar os componentes de software anômala.

Métrica de software é uma medida que diz respeito a quaisquer atributos de qualidade do sistema de software ou processo. É frequentemente impossível medir a atributos de qualidade de software externos, tais como manutenção, compreensibilidade, etc, diretamente. Em tais casos, o atributo externo está relacionado a alguns atributo interno assumindo um relacionamento entre eles e o atributo interno é medido para predizer a característica de software externo. Três condições devem ser espera neste caso:

1. O atributo interno deve ser medido com precisão.
2. Deve existir uma relação entre aquilo que podemos medir e o atributo de comportamento externo.
3. Esta relação tem de ser bem compreendida, foi validada e pode ser expressa em termos de uma fórmula matemática.

O processo de medição

Um processo de medição de software como parte do processo de controlo de qualidade é mostrado na figura abaixo. As etapas do processo de medição são as seguintes:

Selecione medições a efectuar - Selecção das medições que são relevantes para responder as perguntas da avaliação da qualidade.

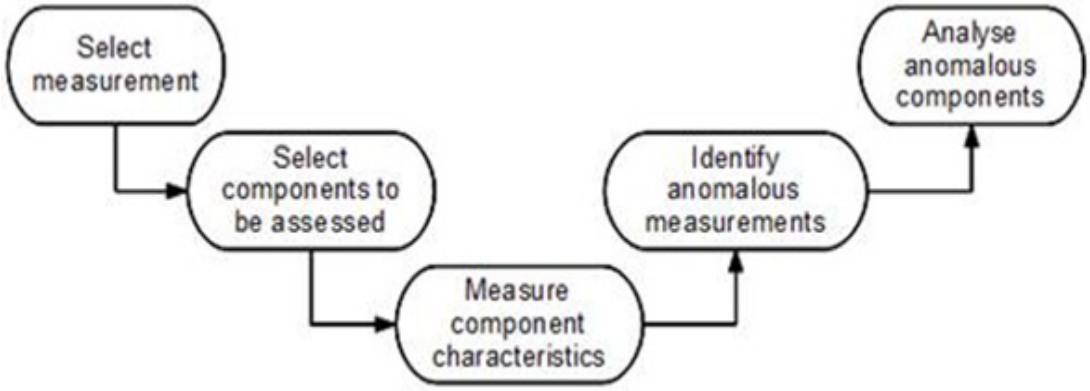
Selecione os componentes a serem avaliados - Seleção de componentes de software a ser medido.

Medir características de componentes - os componentes selecionados são medidos e os valores de métrica de software associada a tomografia.

Identificar as medidas anômala - se qualquer métrica apresentam valores de alta ou baixa significa que o órgão tem problemas.

Analisar os componentes anômala - Se os valores para as métricas de particular anômala têm sido identificados estes componentes têm de ser analisadas para decidir se os valores das métricas anômala significa que a qualidade do órgão está comprometida. Geralmente cada um dos componentes do sistema é analisada separadamente. Medições anômala identificar componentes que podem ter problemas de qualidade.

Figura : o processo de medição de software. Fonte:(internet)



Métricas de produto

As características de software que pode ser facilmente medido como o tamanho não têm uma relação clara e consistente com os atributos de qualidade tais como a compreensibilidade e manutenção. Métricas de produto tem duas classes:

- Métricas dinâmicas - essas métricas (por exemplo o tempo de execução) são medidos durante a execução de um programa.
- Métricas estáticas - métricas estáticas são baseados em medições efetuadas de representações do sistema tais como a concepção, programa ou documentação.
- Métricas dinâmicas podem ser relacionados com a eficiência e a confiabilidade de um programa. Métricas estáticas como o tamanho de código estão relacionados ao software atributos de qualidade tais como a complexidade, compreensibilidade, manutibilidade...

Conclusão

Esta atividade apresentou o controle de qualidade de um mecanismo de acompanhamento do processo de desenvolvimento de software para garantir que os procedimentos de garantia de qualidade e normas estão sendo seguidas. A qualidade de clientes para validação de método a qualidade de um processo ou produto foi destacado. Medição de Software e métricas foram discutidos como instrumentos para avaliar a qualidade do sistema de software atributos ou processo.

Avaliação

1. O que entendes por plano de mediação?
2. Explicar as etapas para o processo de medição de software.

Resumo da unidade

Esta unidade apresentou qualidade de gestão da qualidade e técnicas de gerenciamento. Gestão da qualidade fornece uma verificação independente sobre o software e processo de desenvolvimento de software. Ela garante que produtos finais do projeto são consistentes com os padrões organizacionais e metas. As actividades de gestão da qualidade foram discutidas a garantia de qualidade incluindo planejamento de qualidade e controlo de qualidade.

Avaliação da unidade

Instruções

Responda claramente as questões que se seguem.

Critérios de avaliação

Cada questão equivale a 4 pontos.

Actividades

1. Quais são os dois tipos de normas
2. Desenhe um quadro resumo de normas de produto e de processo.
3. O que é o plano de qualidade?
4. Identificar as principais partes de um plano de qualidade
5. O que entendes por plano de mediação?

RESPOSTA DA UNIDADE

Actividade 1.1

Existem três tipos de padrões de documentação:

a) Normas de processo de documentação - Essas normas definem o processo que deve ser seguido para a produção de documentos.

b) Padrões de documento - Estas normas descrevem a estrutura e apresentação de documentos.

c) Normas para o intercâmbio de documentos - Estas normas garantem que todas as cópias eletrônicas de documentos são compatíveis.

Exemplo de normas

Normas de produto	Normas de processo
Estrutura do documento de requisitos	Processo de aprovação do plano de projeto
Método do formato do cabeçalho	Processo de liberação de versão
Estilo de programação Java	O processo de controle de alterações
Formulário Solicitação de Mudança	Processo de gravação de teste

Actividade 1.2

1. O plano da qualidade define os requisitos de qualidade de software e descreve a forma como estes devem ser avaliadas. O plano de qualidade seleciona os padrões organizacionais que são adequadas a um determinado produto e processo de desenvolvimento.

2. Plano de qualidade tem as seguintes peças:

   a. Introdução do produto.

   b. Planos de produto.

   c. Descrições de processos.

   d. Metas de Qualidade.

   e. Riscos e gestão do risco.

Actividade 1.3

1. Um processo de medição de software como parte do processo de controlo de qualidade é mostrado na figura abaixo. As etapas do processo de medição são as seguintes:

   Selecione medições a efectuar - Selecção das medições que são relevantes para responder as perguntas da avaliação da qualidade.

   Selecione os componentes a serem avaliados - Seleção de componentes de software a ser medido.

   Medir características de componentes - os componentes selecionados são medidos e os valores de métrica de software associada a tomografia.

   Identificar as medidas anômala - se qualquer métrica apresentam valores de alta ou baixa significa que o órgão tem problemas.

   Analisar os componentes anômala - Se os valores para as métricas de particular anômala têm sido identificados estes componentes têm de ser analisadas para decidir se os valores das métricas anômala significa que a qualidade do órgão está comprometida. Geralmente cada um dos componentes do sistema é analisada separadamente. Medições anômala identificar componentes que podem ter problemas de qualidade.

Referências do Curso

Belgamo, A. & Martins, L.E.G., 2000. Estudo Comparativo sobre as técnicas de Elicitação de Requisitos do Software. In XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná.

Guerra, A.C. & Colombo, R.M.T., 2008. Qualidade de Produto de Software. PBQP/MCT. Disponıvel em:< http://www. mct. gov. br/index. php/content/view/2867. html# lista>. Acesso em, 13(09), p.2009.

Jalote, P., 2008. A Concise Introduction to Software Engineering Sétima Edição. Springer, ed.,

Nazareth, D., 1999. Assembling a Metrics Suite for Rule-Based Systems Development. AMCIS 1999 Proceedings, p.23.

Pressman, R.S., 2011. Engenharia de Software - Uma Abordagem Profissional Sétima Edição. M. G. Hill, ed.,

Rocha Balthazar, G. da, 1981. Visão Geral da Qualidade de Software. Revista Eletrônica da Faculdade Metodista Granbery-http://re. granbery. edu. br-ISSN, p.0377.

Santos Bueno, C. de F. dos & Campelo, G.B., Qualidade de Software.

Gerard O'Regan, Introduction to Software Quality, Springer, 2014

Jeff Tian, Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement , IEEE Computer Society, 2005

Darrel Ince, An Introduction to Software Quality Assurance and Its Implementation, McGraw-Hill, 1994

https://en.wikipedia.org/wiki/Problem_frames_approach[Accessed 23/02/2016]

Alistair Sutcliffe, User-Centred Requirements Engineering, Springer-Verlag London Limited 2002

https://en.wikipedia.org/wiki/Requirements_analysis [Accessed 23/02/2016]

David J. Gilmore, Russel L. Winder and Francoise Detienne, User-Centred Requirements for Software Engineering Environments, Springer-Verlag Berlin Heidelberg GmbH, 1994

http://www.rbcs-us.com/documents/Defining-Testing-Article.pdf [accessed 24/02/2016]

http://www.sqa.org.uk/e-learning/SDPL03CD/page_16.htm[accessed 24/02/206]

http://searchsoftwarequality.techtarget.com/definition/conformance-testing[accessed 24/02/2016]

Paul Ammann and Jeff Offutt, INTRODUCTION TO SOFTWARE TESTING, CAMBRIDGE UNIVERSITY PRESS, 2008

William E. Perry, Effective Methods for Software Testing, Third Edition, Wiley Publishing, Inc., Indianapolis, Indiana, 2006

http://www.softwaretestinghelp.com/what-is-conformance-testing/[accessed 24/02/2016]

https://www.smashingmagazine.com/2010/06/design-better-faster-with-rapid-prototyping/ [accessed 24/02/2016]

Avaliação do Curso

Avaliação 1: Questões de todas unidades

Instruções

Responda com clareza as questões que se seguem

Critérios de avaliação

A questão 1 e 2 valem 2.5 pontos cada e a questão 3 a 5 vale 5 pontos, totalizando 20 pontos.

Avaliação

1. Diferencie entre seguintes:
    a. Validação e Verificação;
    b. Quando começar a garantia de qualidade em um projeto?
    c. Qual é a diferença entre Novo teste e testes de regressão?
2. Qual é a diferença entre garantia de qualidade , controlo de qualidade, e testes de software?
3. O que é testware?
4. Qual é o papel da garantia de qualidade em um desenvolvimento do projeto?
5. Qual é o objectivo da estratégia de teste?

Avaliação 2: Questões de unidades 3 e 4

Instruções

Responda com clareza as questões que se seguem

Critérios de avaliação

cada questão vale 5 pontos, totalizando 20 pontos.

http://www.careerride.com/software-quality-assurance-QA-interview-questions.aspx

Avaliação

1. Quais são as diferentes etapas envolvidas na análise orientada a objecto? E porque a análise é uma parte importante na concepção de objectos?
2. Processos de melhoria podem ser aplicados ao processo de desenvolvimento de software, tais como o CMM e o IDEAL. Fale do CMM
3. O que é modelagem de software? Por que utilizar um método para o desenvolvimento de software?
4. O que é Design Centrado no Usuário? Quais são os aspéctos que deve-se tomar em consideração na sua criação?

Respostas do módulo

1.

Verificação e Validação de software é o acto de efetuar testes em tudo que construímos e usamos é algo natural do ser humano, isso porque temos um instinto natural de saber se as coisas funcionam corretamente. Testar tem se mostrado ao longo dos séculos a maneira mais eficiente de verificar se algo está se comportando de forma correta. Produtos de software não fogem a esta regra. Isso porque mesmo o sistema que foi minunciosamente projetado e construído não está livre de apresentar falhas, defeitos e erros. Para que estes problemas sejam descobertos antes do sistema ser entregue para o cliente, executamos uma série de atividades que chamamos de Validação, Verificação.

A actividade de verificação resume-se em responder a esta pergunta. A verificação tem o objetivo de avaliar se o que foi planejado realmente foi realizado. Ou seja, se os requisitos e funcionalidades documentados foram implementados, além disso a verificação também pode ser realizada para especificação de sistemas, para avaliar se os requisitos estão sendo documentados como deveriam e ainda prever falhas ou inconsistências entre requisitos e a validação tem o objetivo de avaliar se o que foi entregue atende as expectativas do cliente. Ou seja, se os requisitos, independente do que foi planejado, estão sendo implementados para atender a regra de negócio do cliente, se o sistema é realmente aquilo que o cliente quer e está pagando para ter. A validação final do sistema é realizada pelo próprio cliente ou usuário.

Um bom momento para iniciar o Garantia de qualidade é a partir do início do arranque do projecto. Isto levará a planejar o processo que irá certificar-se de que o produto que sai atende a expectativa de qualidade ao cliente. QA também desempenha um papel importante na comunicação entre as equipes. Dá tempo para intensificar o ambiente de teste. A fase de testes começa após os planos de teste são escritos, analisados e aprovados.

Novo teste é feito para verificar defeitos correções onde, como regressão é executar para verificar se a correção de defeitos não têm impactado outra funcionalidade que estava trabalhando muito bem antes de fazer alterações no código. - Novo teste está planejado testando com base nas correções de defeitos listados onde, como regressão não é sempre ser específico para qualquer correção do defeito. Também regressão pode ser executado por alguns módulos ou todos os módulos. - Retestagem preocupação com a execução dessas casos de teste que são falharam anteriormente Considerando que a inquietação de regressão com a execução de casos de teste que foi aprovada em versões anteriores. - Novo teste tem maior prioridade sobre a regressão, mas, em alguns casos reteste e testes de regressão são realizadas em paralelo.

Quality Assurance (QA): QA refere-se à forma planejada e sistemática de monitoramento da qualidade do processo que é seguido para produzir um produto de qualidade. QA acompanha os resultados e ajusta o processo para satisfazer as expectativas. Controlo de Qualidade (QC): A preocupação com a qualidade do produto. QC encontra os defeitos e sugere melhorias. O processo estabelecido pelo QA é implementado pelo QC. O QC é da responsabilidade do testador. Teste de software: é o processo de assegurar que o produto que é desenvolvido pelo programador cumpre a exigência do usuário. O motivo para realizar testes é encontrar os erros e certifique-se de que eles são corrigidos.

O subconjunto de software que ajuda na realização do teste de aplicação.

Testware são necessários para planejar, projetar e executar testes. Ele contém documentos, scripts, entradas, resultados esperados, set-up e software adicional ou utilitários usados no teste.

Testware é termo dado a combinação de todas as utilidades e software aplicativo que necessário para testar um pacote de software. Testware é especial porque tem : 1. Finalidade diferente 2. Métricas diferentes para qualidade e 3. usuários diferentes.

QA significa garantia de qualidade. Equipe de QA assegura a qualidade de monitor todo o processo de desenvolvimento. QA acompanha os resultados e processo de ajuste para atender a expectativa. O papel da Garantia de Qualidade é discutido abaixo:-. Equipe de QA é responsável por monitorar o processo a ser realizado para o desenvolvimento - Responsabilidades da equipe de QA está planejando processo de execução de testes. - QA chumbo cria as tabelas de tempo e concorda com um plano de Garantia da Qualidade para o produto. - equipe de QA processo de QA comunicada aos membros da equipe. - equipe de QA garante a rastreabilidade de casos de teste com os requisitos.

Precisamos de Estratégia de Teste pelo seguinte motivo:

- Para ter uma assinado, selado e documento, em que o documento contém detalhes sobre a metodologia de teste, plano de teste e casos de teste entregue.
- Teste documento de estratégia nos diz como o produto de software será testado.
- Documento de estratégia teste ajuda a avaliar o plano de teste com os membros da equipe do projeto.
- Ele descreve as funções, responsabilidades e os recursos necessários para o teste e programação.
- Quando criamos um documento de estratégia de teste, temos de colocar em escrever quaisquer problemas de testes que requerem resolução.
- A estratégia de teste é decidida em primeiro lugar, antes de as decisões de nível inferior são feitas no plano de teste, design de teste, e outras questões de teste.

Avaliação 2: Respostas

Os objectivos ou os passos que estão envolvidos na análise e projeto orientado a objetos são: Criar casos de uso para encontrar as maneiras que um sistema pode interagir com o ambiente. Ele também ajuda a construir uma estrutura global usando os objetos do sistema e permitir fácil de usar modelo a seguir por diante. Identificar os atores e objetos que estão envolvidos: identificando os atores e os objetos com seus nomes de atributos e método facilita a estrutura o sistema de tal forma que um fluxo é criado, que ajudam na concepção do sistema global. estabelecer as relações entre os objetos: estabelecer uma relação entre diferentes objetos permitir que o projecto tornar-se mais clara e utilizável. estabelecer uma interface de cada objecto: permite que o objecto a ser visto em uma posição eo tratamento de exceção também pode ser colocada de forma que se ocorrer algum erro, será conhecido antes de avançar. implementação e teste de objectos tem lugar para ver a boa execução dos objectos e testá-lo para coincidir com os ambientes e needs. Assemble o sistema completo para uso e torná-lo pronto para ser usado pelos usuários.

A análise permite a decomposição de ser feito para construir as peças componentes para ajudar no cálculo do processo e especificar a estrutura do sistema. Isto também permite a definição correcta das funções que são independentes uns dos outros. A análise é feita por decomposição dos módulos ou os componentes. É geralmente realizada em cima para baixo de moda usando a análise estruturada. Este método também é chamado como a decomposição funcional. Esta decomposição funcional combinar com a análise de dados em separado para construir uma meta que é clara e permitir que o sistema a ser definida em seus próprios termos. Análise permite a identificação devem ser realizados para encontrar o processo de negócio, os atores que estão envolvidos na função eo resultado final que tem de ser alcançado.

O Modelo de Maturidade Capacitiva ou simplesmente Capability Maturity Model (CMM) é dividida em cinco níveis:

- Inicial: A organização é caracterizada por um conjunto de actividades adhoc. Os processos não estão definidos e sucesso depende do esforço individual e heroísmo.
- Repetitivo: Neste nível de alguns processos são repetíveis, possivelmente com resultados consistentes.
- Definidos:Neste nível, definimos todos os processos são documentados tanto para a gestão e atividades de engenharia e padrões.
- Gerenciado: medidas detalhadas de cada processo são definidos e dados sobre a qualidade do produto é rotineiramente coletada. Ambos os processos e os produtos são quantitativamente compreendidos e controlados.
- Optimização: Neste otimizamos o aplicativo seguindo processo de melhoria.

Modelagem de software é a atividade de construir modelos que expliquem as características ou o comportamento de um software ou de um sistema de software. Na construção do software os modelos podem ser usados na identificação das características e funcionalidades que o software deverá prover (análise de requisitos), e no planejamento de sua construção. Frequentemente a modelagem de software usa algum tipo de notação gráfica e são apoiados pelo uso de Ferramentas CASE.

A modelagem de software normalmente implica a construção de modelos gráficos que simbolizam os artefatos dos componentes de software utilizados e os seus inter-relacionamentos. Uma forma comum de modelagem de programas procedurais (não orientados a objeto) é através de fluxogramas, enquanto que a modelagem de programas orientados a objeto normalmente usam a linguagem gráfica UML.

Para a modelagem de processos de negócios (BPM) existe uma ferramenta específica denominada BPMN (Business Process Modeling Notation).

Modelagem também é a arte de criar moldes tanto em fundição (neste caso os de areia) como em calçados e em confecção de peças para o vestuario. No caso desta última, este é obtido por uma das três técnicas básicas: moulage, modelagem geométrica ou simples cópia.

Exitem diferentes tipos de modelos para a modelagem dos sistemas tais como Modelos de contexto Modelos de interação, Modelos estruturais, Modelos comportamentais , Engenharia dirigida a modelos

- Os modelos ou a modelagem é uma simplificação da realidade.
- Usámo-lo para compreender melhor o sistema que estamos desenvolvendo.
- Os modelos ajudam a visualizar o sistema como ele é ou como desejamos que seja.
- Os modelos permitem especificar a estrutura ou o comportamento de um sistema.
- Os modelos proporcionam um guia para a construção do sistema.
- Os modelo documentam as decisões tomadas.
- Construímos modelos de sistemas complexos porque não é possível compreendê-los em sua totalidade.

O design centrado no usuário (DCU) parte do princípio que o usuário é o principal foco da realização de um produto, e prioriza as suas necessidades, desejos, expectativas, condicionamentos. Em cada etapa do projeto, designers e projetistas consultam pessoas representativas dos usuários finais, para conhecê-los em profundidade e aliar técnica e sensibilidade para resultados que criem empatia e identidade.

Esta abordagem de projeto é praticada há muito tempo por designers de produtos, e recentemente no design de interação. Da década de 1940 até os anos 80, os sistemas desenvolvidos por engenheiros de software eram operados com base no modo como os computadores funcionavam, especialmente devido às limitações de velocidade de processamento e memória. Além disso, as dificuldades de operacionalização dos equipamentos concentravam a maior parte dos esforços de seu funcionamento.

Os Produtos centrados no usuário geralmente se baseiam nos interesses de segmentos definidos, na medida em que é difícil atender, de maneira personalizada, as necessidades e expectativas de públicos muito amplos. Um moedor a ser utilizado por consumidores de café em grão pode ser melhor desenhado em torno de algumas tipologias claras de determinados grupos de usuários. Já embalagens de café em pó, vendidas para públicos amplos, devem ter características menos personalizadas.

Em projetos centrados nos usuários e realizados em condições ideais, pessoas representativas do público-alvo estão presentes e são ouvidas em todas as etapas, desde as pesquisas iniciais até os testes de usabilidade de protótipos e versões pré-lançamento.

Como são os principais usuários? O que querem ou devem fazer? Como?

Os dados produzidos a partir de entrevistas, pesquisas e testes, sua estruturação em requisitos e análises, ajudam os projetistas a tomar decisões e a delinear os produtos. Os métodos de investigação para a definição dos requisitos incluem estudos etnográficos, análises contextuais, estudos de protótipo, testes de usabilidade, configuração de personas, card sorting, análise de produtos similares pré-existentes, entre outros.

E os métodos de criação de produtos podem também incluir os usuários em seções de design participativo, nas quais processos colaborativos permitem o monitoramento da receptividade e da eficiência do produto durante todo o tempo do projeto.

Resumindo o DCU pressupoe:

- Definição do contexto de uso (quem são os usuários e como usarão o produto)
- Criação de requisitos (comerciais, culturais, funcionais, técnicos) a considerar para que o produto alcance seus objetivos
- Criação de soluções formais e funcionais, que vão do layout de telas e dispositivos até seu desenvolvimento e fabricação.
- Testes e avaliação do produto, com a realização de ajustes (em bases permanentes, antes e depois do lançamento do produto)

Referências do Curso

- Belgamo, A. & Martins, L.E.G., 2000. Estudo Comparativo sobre as técnicas de Elicitação de Requisitos do Software. In XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná.
- Guerra, A.C. & Colombo, R.M.T., 2008. Qualidade de Produto de Software. PBQP/MCT. Disponıvel em:< http://www. mct. gov. br/index. php/content/view/2867. html# lista>. Acesso em, 13(09), p.2009.
- Jalote, P., 2008. A Concise Introduction to Software Engineering Sétima Edição. Springer, ed.,
- Nazareth, D., 1999. Assembling a Metrics Suite for Rule-Based Systems Development. AMCIS 1999 Proceedings, p.23.
- Pressman, R.S., 2011. Engenharia de Software - Uma Abordagem Profissional Sétima Edição. M. G. Hill, ed.,
- Rocha Balthazar, G. da, 1981. Visão Geral da Qualidade de Software. Revista Eletrônica da Faculdade Metodista Granbery-http://re. granbery. edu. br-ISSN, p.0377.
- Santos Bueno, C. de F. dos & Campelo, G.B., Qualidade de Software. Gerard O'Regan, Introduction to Software Quality, Springer, 2014
- Jeff Tian, Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement , IEEE Computer Society, 2005

Sede da Universidade Virtual africana

The African Virtual University
Headquarters

Cape Office Park

Ring Road Kilimani

PO Box 25405-00603

Nairobi, Kenya

Tel: +254 20 25283333

contact@avu.org

oer@avu.org

Escritório Regional da Universidade Virtual Africana em Dakar

Université Virtuelle Africaine

Bureau Régional de l'Afrique de l'Ouest

Sicap Liberté VI Extension

Villa No.8 VDN

B.P. 50609 Dakar, Sénégal

Tel: +221 338670324

bureauregional@avu.org